HYPONUM

API and programming reference

Automation, extension and developer guidance

Desktop edition / October 2026

Public interfaces and internal engineering guidance

Document status

This reference serves two audiences. The first part explains the public Python and command-line interfaces for reproducible analysis. The remaining parts are developer guidance: implementation boundaries, release and support procedures, security, and internal validation policy and evidence. The customer-facing statistical assessment is the separate *Validation* publication.

The desktop product and command-line executable are named HypoNum and `hyponum`, respectively. Saved projects use `.hnum`. Python imports and registered operation identifiers retain `tabelz`; these are distinct interfaces. Historical validation records retain their assessed product name and date. Their conclusions do not automatically apply to the current build.

Statements marked **Plan/Policy** describe intended controls; **Executed evidence** identifies the dated scope actually assessed. Neither internal procedures nor a successful example are a release approval.

Contents

I Python and automation **1**

1 Make a call and read its result **2**

1.1 Two proportions with Fisher's exact test . . . 2
1.2 PCA settings, scores and row identities . . . 3
1.3 Fit a model and score new observations . . . 4
1.4 Import a CSV and execute a table operation . . . 5
1.5 Construct a baseline AnalysisContext . . . 6
1.6 Build and execute a complete recipe . . . 8
1.7 Count regression with an explicit survey design . . . 10
1.8 Fixed-block optimal-design calls . . . 11

2 Python and recipe reference **13**

2.1 Executable project recipes . . . 13
2.2 Operations . . . 13
2.3 Analyses . . . 15
2.4 Baseline methods . . . 16
2.5 Difference Finder and Selection Explainer . . . 18
2.6 Read-only SQL import . . . 19
2.7 Compiling exported reports and preparing support bundles . . . 21
2.8 Upstream software and procedure references . . . 21
2.9 Statistical analysis contracts . . . 22
2.10 Capability interfaces . . . 29
2.11 Measurement-system interfaces . . . 29
2.12 Additional monitoring interfaces . . . 30
2.13 Multivariate monitoring interfaces . . . 31
2.14 Right-censored life distributions . . . 32
2.15 Mixed-censored life distributions . . . 32
2.16 Accelerated life models . . . 32
2.17 Precision planning . . . 32
2.18 Prospective power planning . . . 33
2.19 Stored regression scoring . . . 33
2.20 Stored-model profiling . . . 34

3 Callable reference **35**

3.1 tabelz.analysis . . . 35
3.2 tabelz.data . . . 129
3.3 tabelz.operations . . . 131
3.4 tabelz.project . . . 144
3.5 tabelz.recipes . . . 145
3.6 tabelz.selection . . . 147

4 Settings and returned object fields **148**

4.1 tabelz.analysis . . . 148
4.2 tabelz.contracts . . . 464
4.3 tabelz.data . . . 481
4.4 tabelz.executor . . . 487
4.5 tabelz.operations . . . 487
4.6 tabelz.provenance . . . 491
4.7 tabelz.recipes . . . 495
4.8 tabelz.selection . . . 495

II Extension and implementation **497**

5 API: enterprise contract prototype **498**

5.1 Contract groups . . . 498
5.2 Local mock flow . . . 498
5.3 Explicit non-goals . . . 498

6 Desktop data and display contracts **499**

6.1 Database imports . . . 499
6.2 Derived table lineage . . . 499
6.3 Scatter display allocation and facets . . . 499
6.4 Correspondence-analysis execution . . . 500
6.5 Saved row identities and score columns . . . 500
6.6 Structured Gaussian covariance estimation . . . 500
6.7 Generalized mixed-model numerical policy . . . 501
6.8 GEE numerical conventions . . . 503
6.9 Repeated-covariance numerical conventions . . . 503
6.10 Generalized mixed-model evidence scope . . . 503
6.11 Random-intercept profile details . . . 504
6.12 Factor-analysis boundary verification details . . . 504

6.13 Survival execution and display . . . 504
6.14 Regression numerical dependencies . . . 504
6.15 Nonlinear regression execution and numerical policy . . . 505
6.16 Deming execution and numerical policy . . . 505
6.17 Multiple-imputation numerical policy . . . 506
6.18 Documentation baseline and retained performance context . . . 506

7 Time-series estimation and reconstruction 507

8 Reliability numerical and desktop contracts 509
8.1 Censored log-location-scale fitting . . . 509
8.2 Display and reporting limits . . . 509
8.3 Desktop execution and retained results . . . 510
8.4 Reliability planning duration terminology . . . 510

9 Local analysis plug-in authoring 511
9.1 Authoring a plug-in . . . 511
9.2 Sample and installation . . . 511
9.3 Folder and manifest . . . 511
9.4 Declarative setup form . . . 512
9.5 Run contract and result components . . . 512
9.6 Execution and trust model . . . 514

10 Planning and retained-model numerical contracts 515
10.1 Prospective calculations and transport . . . 515
10.2 Stable scoring covariance and influence . . . 515
10.3 Stored profiler execution . . . 516
10.4 Short-run monitoring transport . . . 516

11 Measurement-system contracts and reference conventions 517
11.1 Standard identity and transport . . . 517
11.2 Execution and retained results . . . 517
11.3 Historical statistical decisions and printed comparisons . . . 517

III Build, release and support 520

12 Release: packaging and release baseline 521
12.1 Repeatable builds . . . 521
12.2 Signing and notarization . . . 522

12.3 Upgrade, rollback, uninstall, and projects . . . 522
12.4 LGPL release obligations . . . 523
12.5 Database driver licences . . . 523
12.6 Corresponding-source preparation for version 0.1.0 . . . 523
12.7 Local crash artifacts . . . 524
12.8 Signed update manifests . . . 524

13 Release: native acceptance protocol 525
13.1 Local Windows checks and clean-machine acceptance . . . 525
13.2 Candidate and disposable test machine . . . 525
13.3 Required physical/native checks . . . 526
13.4 Execution record and release decision . . . 527

14 Release: licensing operations 528
14.1 Product and customer workflow . . . 528
14.2 Issuing, renewing, and rotating keys . . . 528
14.3 Trial and enforcement behavior . . . 529
14.4 Bootstrap dependency license inventory . . . 529
14.5 Open-format reader packaging . . . 530

15 Release: support triage and project recovery 531
15.1 Triage . . . 531
15.2 Recovery . . . 531

16 Release: Qt/PySide corresponding source and replacement 532
16.1 Installation and replacement information . . . 532
16.2 Version 0.1.0 source manifest . . . 532

IV Security and validation evidence 534

17 Security: threat model and privacy baseline 535
17.1 Assets and trust boundaries . . . 535
17.2 High-severity disposition . . . 539
17.3 Process and network policy . . . 539
17.4 Open-format reader boundary . . . 540
17.5 License-key custody . . . 540
17.6 Reproducible validation . . . 540

18 Validation: development checks and integration **542**
18.1 Assign checks to the change . . . 542
18.2 Bound repetition and investigate slow runs . . . 543
18.3 Improve suite runtime without reducing assurance . . . 544

19 Validation: regulated-use master plan **545**
19.1 Executive summary . . . 545
19.2 Start here: first five working days . . . 545
19.3 1. Scope, ownership, and claims . . . 546
19.4 2. Engineering work packages . . . 547
19.5 3. Statistical reference strategy . . . 550
19.6 4. Acceptance and comparison policy . . . 552
19.7 5. Implemented harness and evidence contract . . . 553
19.8 6. End-to-end workflow protocols . . . 554
19.9 7. Regulated-record control assessment . . . 556
19.10 8. Release qualification and approval . . . 556
19.11 9. Handoff packages and ongoing maintenance . . . 557
19.12 Definition of done . . . 557

20 Validation: requirements and risks **558**
20.1 Initial record-control gap assessment . . . 561

21 Validation: numerical policy and record **563**
21.1 Execute and retain evidence . . . 563
21.2 Desktop Help document . . . 563
21.3 Candidate scope and method specification . . . 564
21.4 Comparison policy NIST-LRE-1 . . . 565
21.5 Policy revision and disposition . . . 566
21.6 Next increment . . . 566

22 Statistical reference validation **567**
22.1 Purpose and scope . . . 567
22.2 Evidence register and reproducibility . . . 568
22.3 Statistical process-control charts . . . 576
22.4 Measurement systems, capability and optimization . . . 581
22.5 Distribution diagnostics and simultaneous inference . . . 586
22.6 References and inspected source locations . . . 589

23 Validation: capability workflow **593**
23.1 Scope and intended use . . . 593
23.2 Execute . . . 593
23.3 Independent numeric acceptance . . . 593
23.4 Evidence and limitations . . . 594

24 Validation: industrial ozone workflow **595**
24.1 Intended use and scope . . . 595
24.2 Execute . . . 595
24.3 What the protocol demonstrates . . . 595
24.4 Independent numerical acceptance . . . 596
24.5 Defects identified and changes made . . . 597
24.6 Evidence and limitations . . . 597

25 Validation: time-series workbench **599**
25.1 Scope and contracts . . . 599
25.2 Numerical checks . . . 599
25.3 Execution . . . 600
25.4 2026-09-13 local execution . . . 601
25.5 2026-09-13 bounded-correction execution . . . 601

26 Validation: reliability and performance **602**
26.1 Benchmark foundation and reproducibility rules . . . 602

27 Validation: release evidence expansion **603**
27.1 Acceptance architecture . . . 603
27.2 Retain a complete engineering run . . . 604
27.3 Baseline findings . . . 604
27.4 Native and human follow-up . . . 605

28 Validation: module-foundation record **606**
28.1 Continuation snapshot . . . 606
28.2 Continuation: time-series desktop lifecycle . . . 606
28.3 Continuation: QML shell menu boundary . . . 606
28.4 Continuation: time-series preparation and validation boundaries . . . 607
28.5 Continuation: application-wide report implementation provenance . . . 607
28.6 Continuation: manufacturing services . . . 607
28.7 Continuation: regression service . . . 608
28.8 Scope and structural checks . . . 608

28.9 Environment 608
28.10 Executed checks 608
28.11 Known limitations and baseline failures 610
28.12 Not claimed 610
28.13 Policy disposition for superseded modularization record 610

Part I

Python and automation

CHAPTER 1

Make a call and read its result

The Python package is still named `tabelz`. Use the Python environment in which this edition is installed. Each example below is a complete script: it includes imports, input data, settings, the function call and output access. Save a listing as a `.py` file and run `python filename.py` in that environment. Identical scripts are supplied in `sample-data/programming/`. They require no open desktop window or earlier example variables.

There are three distinct calling conventions. Numerical engines generally accept a Polars table and a typed settings object and return a typed result. The baseline analysis functions accept observation sequences and an `AnalysisContext`, returning both a result and provenance receipt. Operation builders return a command specification; an execution step is needed to apply that specification to data. The examples below show each.

1.1 Two proportions with Fisher's exact test

```
run_hypothesis_test(
    frame: pl.LazyFrame,
    config: HypothesisTestConfig,
) -> HypothesisResult
```

Both arguments are required. `frame` is a lazy Polars table with one row per independent unit. `config.test` selects the procedure; `response` names the outcome column even though the desktop labels its selector *Outcome*. `group` names the two-group column and `success_value` is the success category as a string. The method code for Fisher's test is `"fisher"`. Alpha defaults to 0.05 and the alternative defaults to `"two-sided"`; all relevant choices are explicit here.

```
"""Run directly with Python in an installed Tabelz environment."""

import polars as pl

from tabelz.analysis import HypothesisTestConfig, run_hypothesis_test

data = pl.DataFrame(
    {
        "group": ["A"] * 10 + ["B"] * 10,
        "outcome": ["yes"] * 8 + ["no"] * 2 + ["yes"] + ["no"] * 9,
    }
).lazy()
settings = HypothesisTestConfig(
    test="two_proportion",
    response="outcome",
    group="group",
    success_value="yes",
    method="fisher",
    alternative="two-sided",
    alpha=0.05,
```

```
)
result = run_hypothesis_test(data, settings)
print(result.payload["test_key"])
print(f"p = {result.payload['p_value']:.8f}")
```

Expected output:

```
two_proportion
p = 0.00547749
```

The return is an object, not a bare probability. Its `payload` dictionary (also returned by `to_payload()`) contains the test identity, method, statistic, `p_value`, counts, intervals and result tables. Keys for optional components depend on the selected test. Check those keys before consuming an optional component. Invalid column names, incompatible column types or an unsupported test/method combination raise `ValueError`. This direct call has no desktop exclusions: supply the intended population in `frame` before calling it.

1.2 PCA settings, scores and row identities

```
compute_principal_components(
    frame: pl.LazyFrame,
    config: PcaConfig,
    *,
    row_id_column: str = "__tabelz_row_id",
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> PcaResult
```

`PcaConfig.columns` is a required ordered tuple of numeric column names. `scaling` defaults to `"correlation"`; `"covariance"` retains the original scale. `n_components=None` uses the available component count; this example explicitly retains one. `excluded_row_ids` names stable identities, not row positions. The cancellation callback takes no arguments and returns true to request cancellation.

```
"""A complete correlation-PCA call with retained row alignment."""

import polars as pl

from tabelz.analysis.principal_components import PcaConfig,
    compute_principal_components
from tabelz.selection import with_source_row_ids

data = with_source_row_ids(
    pl.DataFrame({"x": [1.0, 2.0, 3.0, 4.0], "y": [2.0, 4.0, 6.0, 8.0]}).lazy(),
    "pca-example",
)
result = compute_principal_components(
    data,
    PcaConfig(columns=("x", "y"), scaling="correlation", n_components=1),
)
print("means:", result.means)
print("eigenvalues:", tuple(round(x, 6) for x in result.eigenvalues))
print("score rows:", len(result.scores))
print("components:", result.component_count)
assert len(result.row_ids) == len(result.scores) == 4
```

Expected output:

```
means: (2.5, 5.0)
eigenvalues: (2.0, 0.0)
score rows: 4
components: 1
```

Read `result.scores` together with `result.row_ids`: scores describe the analyzed complete cases, not every source position. The result also exposes means, scales, loadings, eigenvalues, variance fractions and excluded or missing counts. Eigenvector signs can reverse without changing the PCA; the example checks eigenvalues rather than imposing an arbitrary sign. Unknown or nonnumeric columns, degenerate scaling inputs and work-limit violations are refused. Settings are strict typed models: unsupported fields or values are rejected during construction.

1.3 Fit a model and score new observations

```
fit_model_scoring(frame, config, *, excluded_row_ids=(), cancel=None)
    -> ModelScoringResult
score_model_scoring(artifact, frame, *, row_ids=None,
                    excluded_row_ids=(), max_rows=100000, cancel=None)
    -> ModelScoringResult
```

The fitting configuration requires `family`, `response` and `predictors`. The default `interval_kind="mean"` requests a mean-response interval; `confidence_level=0.95` is a probability. OLS also accepts `"individual"`. Optional weight, exposure and event fields are conditional on the selected family; they are not interchangeable.

```
"""Fit once, then predict another table without its response column."""

import polars as pl

from tabelz.analysis.model_scoring import (
    ModelScoringConfig,
    fit_model_scoring,
    score_model_scoring,
)

training = pl.DataFrame({"x": [0.0, 1.0, 2.0, 3.0, 4.0], "y": [1.0, 2.0, 2.0, 4.0,
    5.0]})
fitted = fit_model_scoring(
    training,
    ModelScoringConfig(family="ols", response="y", predictors=("x",)),
)
target = pl.DataFrame({"x": [2.0, 5.0, None]})
scored = score_model_scoring(fitted.artifact, target)
print("predictions:", tuple(round(x, 6) for x in scored.predictions))
print("lower:", tuple(round(x, 5) for x in scored.lower))
print("upper:", tuple(round(x, 5) for x in scored.upper))
print("missing rows:", len(scored.missing_row_ids))
assert len(scored.row_ids) == len(scored.predictions) == 2
assert scored.source_count == 3
```

Expected output:

```
predictions: (2.8, 5.8)
lower: (2.06505, 4.07638)
upper: (3.53495, 7.52362)
missing rows: 1
```

The fitted `artifact` contains the frozen coefficients and feature definitions. The scoring result's `row_ids`, `predictions`, `lower` and `upper` align with one another; incomplete and excluded identities are reported separately. Do not zip the two predictions to all three target rows. An unknown category or missing required predictor column raises an error; a missing predictor value is accounted for as an unscored row. The score at $x = 5$ extrapolates beyond this example's training range.

1.4 Import a CSV and execute a table operation

```
SourceImporter().import_csv(path, options, catalog) -> ImportResult
sort(columns, *, descending=False, label=None) -> OperationSpec
FoundationOperationRuntime().execute(frame, spec) -> OperationExecution
```

`path` is a `pathlib.Path`; `options` is a `CsvImportOptions` instance. Its defaults read a UTF-8 comma-separated file with header row 1 and data beginning on row 2. Import registers a table in the supplied `TableCatalog`; access its Polars plan through `result.table.lazy_frame`. The importer supplies stable row identity. Close the catalog when finished with its importer-owned resources.

```
"""Import an explicit small CSV and execute an operation specification."""

from pathlib import Path
from tempfile import TemporaryDirectory

from tabelz.data import CsvImportOptions, SourceImporter, TableCatalog
from tabelz.operations import FoundationOperationRuntime, sort


def main():
    with TemporaryDirectory() as directory:
        path = Path(directory) / "readings.csv"
        path.write_text("name,value\nb,4\na,2\nc,6\n", encoding="utf-8")
        catalog = TableCatalog.empty()
        try:
            imported = SourceImporter().import_csv(path, CsvImportOptions(), catalog)
            spec = sort(["value"], descending=True)
            executed = FoundationOperationRuntime().execute(imported.table.lazy_frame,
     spec)
            print("operation:", spec.operation_type)
            print(executed.frame.select("name", "value").collect().rows())
        finally:
            catalog.close()


if __name__ == "__main__":
    main()
```

Expected output:

```
operation: tabelz.sort
[('c', 6), ('b', 4), ('a', 2)]
```

The `if__name__=="__main__"` guard is necessary when a script invokes the importer's spawned workers on Windows. Keep it when adapting this script. The `sort` builder alone does not modify or collect a table; execution returns a lazy `frame` and a receipt. The final `collect()` materializes the requested columns. Unsupported file settings, malformed input or an unknown operation column are errors, not silently substituted inputs.

1.5 Construct a baseline AnalysisContext

```
descriptives(values, context, *, quantiles=()) -> AnalysisOutcome
```

`values` is an iterable of numbers or `None`. The context declares the population, source fingerprints, operation identity and numerical profile. It is required; `context=...` in a signature is not executable setup. This self-contained example hashes its in-memory input and schema, gives each run fresh identities, and records the installed runtime and Python versions. For imported project data, use that data's actual identities and fingerprints instead of reusing this example's source description.

```
"""Construct the full context required by the provenance-aware baseline API."""

import platform
from hashlib import sha256
from importlib.metadata import version
from uuid import uuid4

from tabelz.analysis import (
    LOCKED_PROFILE_REGISTRY,
    TABELZ_TRANSPARENT_PROFILE_NAME,
    TABELZ_TRANSPARENT_PROFILE_VERSION,
    AnalysisContext,
    FloatValue,
    OperationIdentity,
    RegisteredLineage,
    ScalarComponent,
    create_registered_run_profile,
    descriptives,
)
from tabelz.contracts import EngineVersion, PopulationRef, canonical_json_bytes
from tabelz.provenance import (
    EngineProvenance,
    OperationProvenance,
    ProfileProvenance,
    SourceInput,
)

values = [2.0, 4.0, 6.0]
table_id = str(uuid4())
content = canonical_json_bytes({"value": values})
schema = canonical_json_bytes({"value": "float64"})
profile = create_registered_run_profile(
    LOCKED_PROFILE_REGISTRY,
    TABELZ_TRANSPARENT_PROFILE_NAME,
```

```
    TABELZ_TRANSPARENT_PROFILE_VERSION,
)
context = AnalysisContext(
    population=PopulationRef(
        population_id=str(uuid4()), table_id=table_id, population_name="All three
    readings"
    ),
    input_row_count=len(values),
    sources=(
        SourceInput(
            role="population",
            table_id=table_id,
            content_fingerprint=sha256(content).hexdigest(),
            source_checksum=sha256(content).hexdigest(),
            schema_fingerprint=sha256(schema).hexdigest(),
            row_count=len(values),
        ),
    ),
    operation=OperationProvenance(
        operation_id=str(uuid4()),
        identity=OperationIdentity(
            operation_type="descriptives",
            schema_version="1.0.0",
            implementation_version="1.0.0",
        ),
    ),
    profile=ProfileProvenance(
        profile=profile,
        lineage=RegisteredLineage(
            base_name=profile.name, base_profile_version=profile.profile_version
        ),
    ),
    engine=EngineProvenance(
        engine=EngineVersion(
            engine="tabelz-engine",
            version=version("tabelz"),
            python_version=platform.python_version(),
            adapter="tabelz.analysis.methods",
        ),
        libraries=(),
    ),
)
outcome = descriptives(values, context, quantiles=(0.25, 0.75))
scalars = {
    item.key: item.value.value
    for item in outcome.result.components
    if isinstance(item, ScalarComponent) and isinstance(item.value, FloatValue)
}
for key in ("mean", "median", "stdev", "quantile_0_25", "quantile_0_75"):
    print(f"{key}: {scalars[key]:.1f}")
```

Expected output:

```
mean: 4.0
median: 4.0
stdev: 2.0
```

```
quantile_0_25: 3.0
quantile_0_75: 5.0
```

`AnalysisOutcome.result` is an `AnalysisResult` envelope containing typed scalar, interval, table and warning components; `receipt` is its paired `ProvenanceReceipt`. A scalar value is itself tagged, which is why the example reads `item.value.value`. Inspect `result.status` and warning components: insufficient data can produce a structured result rather than an exception. Inconsistent context row counts, unsupported profile settings and invalid quantile requests raise `ValueError`.

1.6 Build and execute a complete recipe

This example requires the installed runtime's valid recipe-run entitlement. It creates its own source snapshot and bundle, changes the second row's value from 2 to 5, runs the recipe, and writes its output. It needs no pre-existing project or exported script. Save the complete listing and run it with `python recipe_from_scratch.py`. Keep the main guard on Windows.

```
"""Build, execute and write a two-row recipe with a valid run entitlement."""

from datetime import UTC, datetime
from pathlib import Path
from shutil import copyfile
from tempfile import TemporaryDirectory

import polars as pl

from tabelz.contracts.reproduction import SourceProvenance, TableStateRef
from tabelz.operations.table_edits import CELL_EDIT, build_table_edit_spec
from tabelz.project.source_snapshots import write_starting_source_snapshot
from tabelz.recipes import RecipeContext, write_outputs
from tabelz.selection import ROW_ID_COLUMN


def main():
    table_id = "0199a001-0000-7000-8000-000000000001"
    snapshot_id = "0199a001-0000-7000-8000-000000000002"
    source = pl.DataFrame(
        {ROW_ID_COLUMN: ["r1", "r2"], "sample": ["a", "b"], "value": [1, 2]}
    ).lazy()
    with TemporaryDirectory() as directory:
        root = Path(directory)
        snapshot = write_starting_source_snapshot(
            source,
            snapshot_id,
            table_id,
            0,
            SourceProvenance(kind="generated", captured_at=datetime.now(UTC)),
            root,
        )
        bundle = root / "bundle"
        for name, path in snapshot.members.items():
            member = bundle / name
            member.parent.mkdir(parents=True, exist_ok=True)
            copyfile(path, member)
        context = RecipeContext(bundle)
```

```
        starting = context.source(snapshot.metadata)
        edit = build_table_edit_spec(
            CELL_EDIT,
            {"row_id": "r2", "column": "value", "value": 5},
            input_table_id=table_id,
        )
        target = TableStateRef(
            table_id=table_id,
            revision=1,
            origin="operation",
            operation_id=edit.operation_id,
        )
        (edited,) = context.transform(edit, inputs=(starting,), outputs=(target,))
        replay = context.run()
        print("revision:", edited.state.revision)
        print("values:", replay.tables[target.key]["value"].to_list())
        written = write_outputs(replay, root / "output")
        edited_path = root / "output" / f"table-{table_id}-r1.parquet"
        assert edited_path in written
        print("written values:", pl.read_parquet(edited_path)["value"].to_list())
        # Copy files from `written` to your desired destination here if needed;
        # this example's temporary directory is removed on leaving the block.


if __name__ == "__main__":
    main()
```

Expected output with a valid run entitlement:

```
revision: 1
values: [1, 5]
written values: [1, 5]
```

`RecipeContext(bundle)` uses the directory containing the embedded source members. `source(metadata)` returns a `RecipeTableHandle` for the starting revision; it does not import an arbitrary external path. `transform(spec,inputs=(handle,),outputs=(state,))` returns a tuple of handles for the declared output states. The explicit output revision binds the new state to this operation's identity. `run()` returns a `RecipeReplayPayload`; its `tables` mapping is keyed by `TableStateRef.key`, not by a displayed tab name. `write_outputs(payload,destination)` returns paths to all written table and result files. Without a target selection it writes both starting and derived tables, so the example selects revision 1 by its filename. This Python function creates the destination if needed and can overwrite same-named files there; use a fresh directory when retaining earlier runs. The temporary output in this example is removed after printing; choose a new persistent destination when retaining your own outputs.

An unavailable run entitlement raises `RecipeEntitlementError`. Missing or changed source members, incompatible operation versions and inconsistent input/output states are rejected rather than repaired by a different computation. These failures occur before a valid output can be claimed.

The next chapter explains recipes and the method contracts. The callable catalog then gives exact imports, signatures and defaults, followed by the fields of the settings and result types. Internal engineering chapters are separate from these public invocation instructions.

1.7 Count regression with an explicit survey design

```
compute_survey_count(frame, config, *, excluded_row_ids=(), cancel=None)
    -> SurveyCountResult
```

The input is an eager or lazy Polars table. Required settings are the saved design and response column; this example also names one numeric covariate. Supply the design explicitly in a direct Python call: an open desktop table is not consulted. `weight`, first-stage `strata` and `units` name columns. With no FPC or second stage, this example uses the ultimate-cluster variance. Its 24 observations have two strata and eight PSUs. No domain or row exclusions are requested.

Save the following supplied script at its stated location and run `pythonsample-data/programming/survey_count.py` from the repository root. The script resolves its CSV relative to its own location; it also works when invoked from another directory. The CSV is shipped at `sample-data/manual/survey-counts-twenty-four.csv`.

```
"""Run from any working directory using the installed HypoNum environment."""

from pathlib import Path

import polars as pl

from tabelz.analysis.survey_count_contracts import SurveyCountConfig
from tabelz.analysis.survey_count_models import compute_survey_count
from tabelz.contracts.survey_design import SurveyDesign, SurveyStage

path = Path(__file__).resolve().parents[1] / "manual/survey-counts-twenty-four.csv"
frame = pl.read_csv(path)
design = SurveyDesign(weight="w", stage1=SurveyStage(strata="h", units="psu"))
config = SurveyCountConfig(
    design=design,
    response="y",
    covariates=("x",),
    factors=(),
    family="poisson",
    exposure=None,
    inference="design_df",
    confidence_level=0.95,
)
result = compute_survey_count(frame, config)
for coefficient in result.fit.coefficients:
    print(coefficient.term, f"{coefficient.estimate:.6f}")
print("rows:", result.fit.rows_used)
print("test df:", result.fit.test_df)
```

Expected output, rounded to six decimal places:

```
(Intercept) 0.296335
x 1.265942
rows: 24
test df: 5
```

`result.fit.coefficients` contains term, estimate, standard error, statistic, degrees of freedom, probability and interval fields. `result.fit.design` records the design summary. The result is not an ordinary independent-observation Poisson covariance calculation.

To select NB2, use `family="nb2_fixed"` with a positive externally supplied `dispersion`, or `family="nb2_estimated"` without that field. The latter returns the jointly fitted dispersion and its uncertainty fields; they are absent for Poisson. `exposure` names a strictly positive column whose log is an offset, not a fitted predictor. Domains use `domain_column` and `domain_values` together and retain other rows in the variance design. Row exclusions, invalid counts or exposure, unsupported designs and failed fits are refused; do not filter the table first to imitate a survey domain. Full defaults and result types appear in the callable catalog.

1.8 Fixed-block optimal-design calls

The direct design generator has this signature:

```
from tabelz.analysis.doe_designs import generate_catalog_design
from tabelz.analysis.doe_types import DesignCatalogConfig

generate_catalog_design(config: DesignCatalogConfig, *,
                        cancel: Callable[[], bool] | None = None
                        ) -> GeneratedDesign
```

`cancel` is an optional callback returning true to stop the work; cancellation raises `OptimalDesignCancelledError`. Ordinary callers can omit it. The returned design has a Polars `frame` of run settings and a `diagnostics` mapping with the achieved criterion, rank, capacities, search work and stopping reason.

This complete example requires no input file:

```
from tabelz.analysis.doe_designs import generate_catalog_design
from tabelz.analysis.doe_types import DesignCatalogConfig

config = DesignCatalogConfig.from_dict({
    "family": "optimal",
    "factors": [{"name": n, "low": -1, "high": 1}
                for n in ("A", "B")],
    "optimal_runs": 9, "blocks": 2,
    "planned_model": "main_effects",
    "optimal_criterion": "D",
    "optimal_planning_version": "1.1.0",
    "randomize": False,
})
result = generate_catalog_design(config)
print(result.frame.height)
print(result.diagnostics["block_capacities"])
print(round(result.diagnostics["treatment_log_determinant"], 6))
print(result.diagnostics["search_stopping_reason"])
```

Expected output:

```
9
[5, 4]
4.341205
no_improving_coordinate
```

The feasible grid here is $\{-1, 0, 1\}^2$. The combined fixed-block and main-effects model has rank 4. Its treatment log determinant improves from 2.261763 to 4.341205 after two sweeps and 162 candidate evaluations. The result explicitly retains `global_optimum_certified=False`. Choose

`optimal_criterion="I"` to minimize the equal-block-average prediction variance over the feasible grid; this is not prediction in a new random block. Unsupported rank or constraints raise a refusal rather than reducing the required model.

Prospective planning exposes the same optional cancellation keyword:

```
from tabelz.analysis.guided_doe_planning import plan_candidate

plan_candidate(payload: Mapping[str, Any], objective: str,
               spec: ProspectiveContrast, *,
               cancel: Callable[[], bool] | None = None
               ) -> ProspectiveResult
```

The payload is the design configuration, `objective` identifies the planning task, and `spec` is the declared prospective contrast and noise/effect specification. New candidate summaries may contain `optimal_search`; older/default summaries omit that optional metadata. New guided calls record `planning_version="1.1.0"`; old or untagged calls retain their earlier selection and serialization conventions.

CHAPTER 2

Python and recipe reference

This chapter is for readers who automate work through Python or the command line. Recorded workflow actions use stable identifiers. Operation IDs are the same in Python, in a saved workflow and in a recipe; the menu label does not create a second identifier. The tables below map a baseline subset of operations and analyses to Python entry points. The complete callable catalog in chapter 3 provides import paths, signatures, settings fields and returned object fields, including the direct fitted-model interfaces. Registered recipe actions also run through the versioned recipe API described next; an action ID is not necessarily a Python function name.

2.1 Executable project recipes

With a compatible installed HypoNum runtime and valid run entitlement, the following commands reproduce a new-format project or explicitly run an exported Python script. The output directory must not already exist:

```
hyponum recipe run study.hnum --output reproduced
hyponum recipe run study.hnum --target result-<stable-id> --output one-result
hyponum script run exported/analysis.py --output script-results
```

The first command validates typed project members and runs registered engine operations. The second runs only the selected target and its transitive prerequisites. The script command deliberately executes user-controlled Python in a separate process: review edited exports and their data first. Neither command needs a source checkout or development tools. An installed runtime must provide the same registered operation versions and numerical dependencies; required plugins must be installed separately with their compatible identity. Opening or viewing a project does not run its script.

Generated `analysis.py` is readable Python built from the saved workflow. It calls `tabelz.recipes.RecipeContext.source` for each embedded starting table, `transform` or `analyze` for recorded operations, and `run` followed by `write_outputs`. The public handles carry exact table revisions and result identities. Applications can build their own recipes from explicit, versioned `OperationSpec` and `AnalysisRecipeRequest` values using the same API; the engine rejects an unregistered implementation. Editing an exported script makes an independent Python artifact. HypoNum does not convert arbitrary Python changes back to a desktop workflow, and an exported script still requires HypoNum to execute.

2.2 Operations

Operation ID	User-facing purpose	Python entry point
`tabelz.concatenate`	Combine compatible tables.	`tabelz.operations.concatenate`

tabelz.computed_column	Add a declared derived column.	tabelz.operations.computed_column
tabelz.principal_component_scores	Add scores from a retained PCA basis and source-row population.	tabelz.operations.principal_component_scores
tabelz.factor_scores	Add scores from a retained factor-analysis basis and source-row population.	tabelz.operations.factor_scores
tabelz.hierarchical_cluster_labels	Add cluster labels from a retained dendrogram cut and clustering population.	tabelz.operations.hierarchical_cluster_ labels
tabelz.kmeans_cluster_labels	Add cluster labels from a retained k-means assignment and clustering population without refitting.	tabelz.operations.kmeans_ cluster_ labels
tabelz.item_analysis_scores	Add a mean or sum scale score from a retained item design and source-row population.	tabelz.operations.item_analysis_scores
tabelz.model_scoring_scores	Add response-scale predictions and optional interval bounds from a frozen regression artifact.	tabelz.operations.model_scoring.model_scoring_scores
tabelz.classification_tree_scores	Add predicted classes from a retained safe tree and declared scoring population.	tabelz.operations.classification_tree_scores
tabelz.regression_tree_scores	Add predictions from a retained safe regression tree.	tabelz.operations. regression_tree_scores
tabelz.reshape_stack	Stack declared wide columns with source-row lineage.	tabelz.operations.reshape. reshape_stack
tabelz.reshape_pivot	Pivot declared key, name and value columns with explicit duplicate aggregation.	tabelz.operations.reshape. reshape_pivot
tabelz.deduplicate	Remove duplicate rows by declared keys.	tabelz.operations.build_ deduplicate
tabelz.drop_columns	Remove selected columns.	tabelz.operations.drop
tabelz.filter	Keep rows matching a declared expression.	tabelz.operations.build_ filter
tabelz.formula_column	Replace a column with a bounded numeric or text formula.	tabelz.operations.formula_column
tabelz.group_summary	Aggregate groups with declared aggregations.	tabelz.operations.build_group_ summary
tabelz.join	Join tables with cardinality diagnostics.	tabelz.operations.join
tabelz.johnson_transform	Add standard-normal scores using retained Johnson parameters without refitting.	tabelz.operations.johnson_transform.johnson_transform

`tabelz.study_worksheet`	Create a randomized measurement study worksheet without an input table.	`tabelz.operations.study_worksheet. execute_study_worksheet`
`tabelz.random_table`	Create a new table of seeded draws from a declared distribution without an input table.	`tabelz.operations.random_data. random_table_operation`
`tabelz.keep_per_group`	Keep a bounded selection per group.	`tabelz.operations. keep_per_group`
`tabelz.recode`	Recode declared values.	`tabelz.operations. build_ recode`
`tabelz.rename_columns`	Rename columns.	`tabelz.operations. rename`
`tabelz.reorder_columns`	Reorder columns.	`tabelz.operations. reorder`
`tabelz.row_subset`	Keep a row range.	`tabelz.operations. row_subset`
`tabelz.sample`	Take a deterministic sample.	`tabelz.operations. sample`
`tabelz.select_columns`	Keep selected columns.	`tabelz.operations. select`
`tabelz.sort`	Sort rows by declared columns.	`tabelz.operations. sort`

2.3 Analyses

Analysis ID	User-facing purpose	Python entry point
`descriptives`	Counts and descriptive statistics.	`tabelz.analysis. descriptives`
`mean_ confidence_ interval`	Mean confidence interval.	`tabelz.analysis. mean_ confidence_ interval`
`pearson_correlation`	Linear correlation.	`tabelz.analysis. pearson_ correlation`
`spearman_correlation`	Rank correlation.	`tabelz.analysis. spearman_ correlation`
`ordinary_least_squares`	Linear regression.	`tabelz.analysis. ordinary_ least_ squares`
`one_way_anova`	Group mean comparison.	`tabelz.analysis. one_way_ anova`
`normal_capability`	Normal capability estimate.	`tabelz.analysis. normal_ capability`
`two_population_ numeric_ comparison`	Numeric population comparison.	`tabelz.analysis. two_population_ numeric_ comparison`

`two_population_ categorical_ comparison`	Categorical population comparison.	`tabelz.analysis. two_population_ categorical_ comparison`
`difference_finder`	Rank declared population differences.	`tabelz.analysis. find_ differences`
`selection_explainer`	Describe predictors associated with a selection; not causal.	`tabelz.analysis. explain_ selection`

Every registered analysis call in the table requires an `AnalysisContext`. A call returns typed result components together with a provenance receipt. Population identifiers, options, and receipts are part of a reproducible invocation rather than display-only metadata.

2.4 Baseline methods

These methods return an `AnalysisOutcome` containing an `AnalysisResult` and a reconciled `ProvenanceReceipt`. Every method accepts an `AnalysisContext`. Calculations use only Python's standard library, so no third-party numeric library is recorded. Null and non-finite values are excluded before calculation. Exclusion counts are disjoint and are applied in this order: filter, null, then non-finite.

The behavior below documents a transparent implementation contract; it is not a claim of conformity to a named external standard. Profile assumptions are explicit. Unsupported input warns or fails and never silently substitutes a different model.

2.4.1 Descriptive statistics

`descriptives(values, context,quantiles=())` returns counts, arithmetic mean, median, extrema, requested quantiles, and a profile-selected standard deviation. Quantiles are unique finite floats in $[0, 1]$. For a sorted sample of size n, the linear-interpolation position is $h = (n-1)q$ and the value is

$$x_{\lfloor h \rfloor} + (h - \lfloor h \rfloor)\left(x_{\lceil h \rceil} - x_{\lfloor h \rfloor}\right).$$

No finite values is insufficient data. If the standard deviation is undefined, it is omitted rather than represented by a non-finite number.

2.4.2 Mean confidence interval

`mean_confidence_interval(values,context)` reports

$$\bar{x} \pm z_{1-\alpha/2}\, s/\sqrt{n}.$$

It assumes independent observations and uses the normal critical-value approximation configured by the active profile. At least two finite observations are required.

2.4.3 Pearson and Spearman correlation

`pearson_correlation(x,y, context)` computes covariance divided by the product of the centered sums of squares. `spearman_correlation(x,y, context)` first replaces each input with average ranks (including average ranks for ties), then applies Pearson correlation. Both methods use pairwise complete rows. A constant input makes the correlation undefined.

2.4.4 Ordinary least squares

`ordinary_least_squares(x,y,context)` fits the one-predictor model

$$y = b_0 + b_1 x, \qquad b_1 = S_{xy}/S_{xx}, \qquad b_0 = \bar{y} - b_1 \bar{x}.$$

The intercept is included. Inputs are listwise complete, and R^2 is reported when defined. A constant predictor raises the singular error selected by the profile; a constant response causes R^2 to be omitted.

2.4.5 One-way ANOVA

`one_way_anova(groups,context)` reports the fixed-effects statistic

$$F = \frac{SS_{\text{between}}/(k-1)}{SS_{\text{within}}/(n-k)}.$$

It requires two nonempty groups and positive within-group variation. For inferential interpretation, the documented assumptions are independent observations, approximately normal within-group errors, and comparable variance.

2.4.6 Normal capability

`normal_capability(values,lower_spec,upper_spec,context)` reports

$$C_p = \frac{USL - LSL}{6s}, \qquad C_{pk} = \min\left(\frac{USL - \bar{x}}{3s}, \frac{\bar{x} - LSL}{3s}\right).$$

The process is assumed stable and approximately normal, with meaningful specification limits. Limits must be finite and ordered, and the sample deviation must be nonzero.

2.4.7 Two-population comparisons

Numeric comparison

`two_population_numeric_comparison(first,second,context)` reports an independent-sample mean difference and Cohen's d using the pooled standard deviation. Each population needs at least two finite observations, and the pooled deviation must be nonzero.

Categorical comparison

`two_population_categorical_comparison(first,second,context)` computes the Pearson chi-square statistic on a two-by-k contingency table. It emits no p-value. Null categories are excluded; at least two levels and nonempty populations are required.

2.4.8 References and numeric policy

The implementation references the NIST/SEMATECH *e-Handbook of Statistical Methods*, chapters on descriptive statistics, correlation, regression, ANOVA, and process capability, and Wasserman, *All of Statistics*, sections on least squares and contingency tables. Golden numeric comparisons use relative tolerance 10^{-9} and absolute tolerance 10^{-12}. The default `tabelz-transparent` profile makes no AIAG, VDA, ISO, or ASTM conformity claim.

2.5 Difference Finder and Selection Explainer

Stable source-row identity is retained through filters, sorts, joins, exclusions, plots, reports, and plugin inputs. Excluded rows remain visible in the source view but are removed from the shared analysis source. When identities are regenerated, exclusions persist by current row position. Sampling and plot data are bounded and disclose source, valid, and rendered counts.

2.5.1 Difference Finder MVP

`find_differences(first, second, context=..., options=..., specifications=...)` compares two explicit populations and returns the standard `AnalysisOutcome`. Its `findings` table contains ranked values, differences, metric-specific effect sizes, counts, scores, and drill references. Its `significance` table contains only tested rows; each such row has a p-value, a Benjamini–Hochberg-adjusted p-value, an effect size, and both sample counts.

For numeric metrics, the finite signed normalized delta is

$$\frac{first - second}{|first| + |second|},$$

with value zero when both inputs are zero. This identifies constant shifts without an unbounded zero-variance standardization. Spread is the difference of sample standard deviations. Quantiles use the linear interpolation defined in the baseline-method contract. Missingness and out-of-specification metrics are rates; categorical rows are level-proportion differences.

Non-finite numeric values and `None` are missing, and valid numeric calculations exclude them. Independent p-values use normal approximations and are corrected by Benjamini–Hochberg over all tested rows. No p-value is emitted for spread or quantile rows, or for unsupported paired proportions.

Options, pairing, and provenance

`DifferenceFinderOptions` is versioned. It records rank weights, multiplicity policy, quantiles, the minimum meaningful raw metric difference, maximum output size, and (in paired mode) an existing `EntityKey`. Paired inputs require matching, unique, non-null composite key sets and are aligned by that key. Correlated-signal clustering is explicitly deferred from this MVP.

The provenance receipt records both population identifiers, the exact options JSON, the ranking fingerprint, and canonical exact specification snapshots. `render_difference_finder_python` produces replay source that invokes the same public method with the caller's populated `context`.

2.5.2 Selection Explainer

`explain_selection(selected, comparison, context=..., options=...)` explains one explicit population relative to another. Both identifiers are required, distinct, bounded, and recorded in the receipt. The result is an typed `AnalysisOutcome` with typed `explanations` and `split_evidence` tables.

Every result carries warning code `analysis.selection_explainer.not_causation` and machine label `explanation_not_causation`. Rows describe associations only: this MVP makes no causal, multivariate, or general predictive claim.

Numeric separation is the finite mean difference normalized by the two mean magnitudes. Categorical enrichment is a type-tagged level-proportion difference. `None` and non-finite numeric

values are excluded from descriptive values, with exclusion counts exposed. A declared time field emits bounded calendar-day buckets (ISO-8601 values must include an offset and are normalized to UTC) or numeric-floor buckets. Only the highest absolute differences, then lexical ties, up to `max_time_buckets`, are shown.

Threshold evidence

Threshold rows use finite values only. Midpoints and both directions are selected from the training split, then ranked by held-out balanced accuracy times stability, where

$$stability = 1 - |training_score - validation_score|.$$

`max_threshold_candidates` bounds candidate evaluation. Training and validation finite sample counts are persisted with each threshold. A dataset that cannot supply both populations in both splits is rejected.

Deterministic split and leakage checks

Without declarations, the deterministic row split is labelled `not_declared` for both entity and time leakage. With an `EntityKey`, all rows for one entity stay in one split, and an entity occurring in both populations is rejected. With a declared time field, validation is a chronological tail; training must precede the first validation time globally across both populations. Combined entity/time mode rejects a split when whole-entity grouping would cross that boundary.

Successful checks are labelled `checked_grouped` and `checked_chronological`. Split rows use the caller's explicit population identifiers. A time field cannot mix numeric and ISO representations.

Options, CLI, and replay

Options are strict and versioned as `1.0.0`. They contain validation fraction, threshold/time/explanation limits, rank weights, an optional time field, and an optional entity key. Their canonical JSON and ranking fingerprint are receipt parameters. The headless command accepts an invocation file:

```
hyponum explain-selection invocation.json
```

`render_selection_explainer_python` recreates the same invocation when given the two populations and the caller's `context`.

References for Difference Finder include Cohen (1988), *Statistical Power Analysis*, and Benjamini and Hochberg (1995), JRSS B 57.

2.6 Read-only SQL import

`SqlImportOptions` and `SourceImporter` expose the same SQL query path used by Desktop for local **SQLite** and **DuckDB** databases. The dialect is selected by `database_kind` (`"sqlite"` or `"duckdb"`); when it is omitted, `.duckdb` and `.ddb` files are DuckDB and `.sqlite`, `.db`, and `.sqlite3` files are SQLite. PostgreSQL, SQL Server, credential storage, and database writes are outside this increment.

2.6.1 Python API

The importer accepts a path, options, and a table catalog. A typical parameterized SQLite read is:

```
from pathlib import Path

from tabelz.data import SqlImportOptions, SourceImporter, TableCatalog

catalog = TableCatalog.empty()
result = SourceImporter().import_sql(
    Path("quality.sqlite"),
    SqlImportOptions(
        query=(
            "SELECT sample_id, measured_at, reading "
            "FROM measurements WHERE measured_at >= :start_date"
        ),
        parameters={"start_date": "2026-01-01"},
        dataset_name="Current measurements",
        stable_key_columns=("sample_id",),
        data_types={"measured_at": "datetime"},
    ),
    catalog,
)
frame = result.table.lazy_frame
```

The same import against a DuckDB database uses `\$name` placeholders:

```
result = SourceImporter().import_sql(
    Path("quality.duckdb"),
    SqlImportOptions(
        database_kind="duckdb",
        query=(
            "SELECT sample_id, measured_at, reading "
            "FROM measurements WHERE measured_at >= $start_date"
        ),
        parameters={"start_date": "2026-01-01"},
        dataset_name="Current measurements",
        stable_key_columns=("sample_id",),
        data_types={"measured_at": "datetime"},
    ),
    catalog,
)
```

Use `preview_sql()` for a bounded preview or `list_sql_tables()` to enumerate non-system tables and views. `list_sql_tables()` detects the dialect from the file suffix; pass `kind=` to force one. Catalog entries identify object kind, declared columns, ordered primary-key columns, nullability, and suggested HypoNum types.

2.6.2 Headless command

The equivalent command writes CSV or Parquet. The dialect is inferred from the database file suffix; pass `--database-kind` to force it:

```
hyponum sql-import quality.sqlite \
  --query "SELECT sample_id, reading FROM measurements WHERE reading >= :minimum" \
```

```
  --parameter minimum=2.5 \
  --key sample_id \
  --type measured_at=datetime \
  --output readings.parquet

hyponum sql-import quality.duckdb \
  --database-kind duckdb \
  --query "SELECT sample_id, reading FROM measurements WHERE reading >= $minimum" \
  --parameter minimum=2.5 \
  --key sample_id \
  --output readings.parquet
```

The printed JSON record includes `source_type` with the resolved dialect.

2.7 Compiling exported reports and preparing support bundles

Exported report LaTeX uses `booktabs`, `longtable`, `pdflscape` and `pgfplots`. Compile it with a compatible TeX distribution after reviewing the generated file.

A local support bundle can be created with:

```
hyponum diagnostics bundle --output hyponum-support.zip --log app.log --receipt
    receipt.json
```

This writes a local archive; it does not upload it. Project, source and cache payloads are excluded, and common credentials/private paths are redacted. Inspect the contents before sharing them.

2.8 Upstream software and procedure references

These references describe broader library or procedure behavior. HypoNum may expose a subset with different defaults. Use the preceding HypoNum contracts and desktop chapters for the supported interface.

1. Polars project, *Python API Reference*.
   Official reference.
2. Polars project, *DataFrame.join*.
   Official reference.
3. Polars project, *DataFrame.unique*.
   Official reference.
4. scikit-learn project, *KNNImputer*.
   Official reference.
5. NumPy project, *Statistics*.
   Official reference.
6. SciPy project, *Statistical functions*.
   Official reference.
7. statsmodels project, *Generalized Linear Models*.
   Official reference.
8. statsmodels project, *Power and sample size calculations*.
   Official reference.
9. SciPy project, *Shapiro–Wilk test*.
   Official reference.

10. statsmodels project, *Fitted-normal Anderson–Darling diagnostic*.
Official reference.
11. NIST/SEMATECH, *Grubbs' test for outliers*.
Official reference.
12. R project, *Exact Poisson tests*.
Official reference.
13. SciPy project, *Dunnett multiple comparisons against a control*.
Official reference.
14. SciPy project, *Studentized range distribution*.
Official reference.
15. R project, *Friedman rank sum test*.
Official reference.
16. statsmodels project, *OLS influence diagnostics*.
Official reference.
17. R project, *Prediction for linear models*.
Official reference.
18. Automotive Industry Action Group, *Measurement Systems Analysis Reference Manual*, 4th edition, 2010. Cited by section and page in the Gage R&R chapter; no text or tables are reproduced.
19. D. J. Wheeler and R. W. Lyday, *Evaluating the Measurement Process*, 2nd edition, SPC Press, 1989 (source of the Chapter IV Section E worksheet).
20. ASTM E691, *Standard Practice for Conducting an Interlaboratory Study to Determine the Precision of a Test Method* (pooled standard deviation study and the h and k statistics).
21. A. J. Duncan, *Quality Control and Industrial Statistics*, 5th edition, Irwin, 1986 (d_2^* approximation).
22. Minitab, *Example of Crossed Gage R&R Study* (help page; cross-check of the ANOVA example).

2.9 Statistical analysis contracts

2.9.1 Panel fixed effects and mixed structures

`tabelz.analysis.panel_fixed_effects.compute_panel_fixed_effects` takes an explicit frame, frozen `PanelFixedEffectsConfig`, row identities and exclusions. `PanelFixedEffectsResult` separates within R^2, slope uncertainty, entity/time coverage and row-aligned output. Schemas are `panel-fixed-effects-config` and `panel-fixed-effects-result`. The CR1 correction counts all G entity intercepts and k slopes; it is not inferred from the demeaned design's column count.

`tabelz.analysis.mixed_model_structures.compute_mixed_model_structures` accepts `MixedModelStructuresConfig`, frame, identities and exclusions. It supports one correlated random intercept/slope component or two to three explicitly crossed/nested independent components. The result retains fixed inference, covariance/components, boundary/convergence status and UUID-keyed population/conditional predictions. ML alone provides AIC. Schemas are `mixed-model-component`, `mixed-model-structures-config` and `mixed-model-structures-result`.

The direct numerical entry points accept an explicit frame, typed settings, stable string row identities, exclusions and optional cancellation. Their result populations must be interpreted using the returned accounting, rather than the current desktop selection.

2.9.2 Correspondence analysis

`analysis.correspondence_analysis` exposes `CorrespondenceConfig`, `CorrespondenceResult` and `run_correspondence`. The two input modes are categorical roles with optional exact integer frequencies, and integer count columns with explicit row labels. Exclusions and cancellation are explicit; outputs are finite strict JSON. Schemas are `correspondence-analysis-config` and `correspondence-analysis-result`. The method, implementation version, application version, schema version and Git revision are distinct identities.

2.9.3 Principal components and factor analysis

`tabelz.analysis.principal_components.compute_principal_components` takes a frozen `PcaConfig`. `PcaResult` retains row-aligned scores, unit-eigenvector loadings, complete-case means and scales, all available eigenvalues and variance fractions, component count, rank and omission counts. The schema names are `pca-config` and `pca-result`. Correlation scaling is the default; covariance scaling is explicit. The retained `tabelz.principal_component_scores` operation (implementation 1.0.0) stores the basis and population and does not refit.

`tabelz.analysis.factor_analysis.compute_factor_analysis` takes `FaConfig` with measurement columns, explicit factor count, optional orthogonal varimax rotation and bounded work limits. `FaResult` preserves displayed and unrotated loadings, communalities, uniquenesses, correlation eigenvalues, regression score weights, aligned scores, means/scales and omission counts. Status is `converged`, `nonconverged` or `heywood`; the latter two provide no scores. Schemas are `fa-config` and `fa-result`.

`tabelz.factor_scores` implementation 1.0.0 retains names, source identity, means/scales, displayed-frame Thurstone regression weights and fit-population IDs. It projects those rows without refitting and leaves omitted rows missing. Existing names are rejected on creation. Replay against already-created columns requires agreement with the retained basis and rejects conflicts.

2.9.4 MANOVA

`tabelz.analysis.manova.compute_manova` accepts a lazy or eager frame, frozen `ManovaConfig`, exclusions and stable row identities. `ManovaResult` retains Pillai/Wilks term tests and labelled F approximations, Holm-adjusted response follow-ups, residual covariance, design/residual ranks, counts and bounded effect data. Schemas are `manova-config` and `manova-result`. The shared general-linear encoder supplies explicit Type II/III hypothesis matrices.

2.9.5 Validated regression contracts

`tabelz.analysis.predictive_validation.compute_validated_regression` takes `ValidatedRegressionConfig`: one response, numeric/categorical predictors, algorithm, predictor imputation, seed, test fraction and limits. `SplitAssignment` retains the seeded partition. `ValidatedRegressionResult` retains row-ID split/fold assignments, population accounting, original-scale coefficients, training fill values, category maps, scaler statistics, tuning and test metrics, and bounded display points. Schemas are `validated-regression-config` and `validated-regression-result`. The separate Fit Y by X workflow retains its historical method identity and preprocessing conventions.

2.9.6 Generalized mixed models

The random-intercept GLMM method identity is `models.glmm_random_intercept`; schema entries are `glmm-config` and `glmm-result`. Saved summaries retain the full configuration,

including response mapping, grouping, accuracy settings and population, for explicit rerun. Numerical certification policy is in the engineering corpus.

2.9.7 Repeated residual covariance

`tabelz.analysis.repeated_covariance.compute_repeated_covariance` provides schemas `repeated-covariance-config` and `repeated-covariance-result`. Programmatic requests may additionally name nested fixed-effect columns for an ML comparison on identical rows and covariance structure. Output includes both ML log likelihoods, likelihood-ratio statistic and degrees-of-freedom difference, AIC and BIC, without a p-value: nesting alone does not establish an interior chi-square reference at a covariance boundary. The desktop does not automatically select models or compare REML across different fixed designs.

2.9.8 Kaplan–Meier and log-rank contracts

`tabelz.analysis.survival_nonparametric.compute_survival` takes `SurvivalConfig`: time/status/group roles, explicit event and censor values, confidence level and row limit. `SurvivalResult` retains an exact `SurvivalCurve` per group with times, pre-time risk sets, event and censor counts, survival, Greenwood standard errors and pointwise log-log limits. Unavailable quantities are null, with their status.

`MedianSurvival` distinguishes estimated, not-reached and unavailable quantities. `LogRankTest` contains the two-group calculation or its unavailability reason. Source/excluded/missing/analyzed counts and the analyzed-row SHA-256 fingerprint are retained. Schemas are `survival-config` and `survival-result`. Validators reject rising curves, censor-only drops, inconsistent risk sets and unexplained unavailable tests. The method identity is `survival.kaplan_meier_log_rank`.

2.9.9 Nonlinear regression

`tabelz.analysis.nonlinear_regression` exports `NonlinearRegressionConfig`, `NonlinearRegressionArtifact`, `NonlinearRegressionResult`, `fit_nonlinear` and `predict_nonlinear`. Immutable strict contracts are defined in `analysis.nonlinear_contracts`; schema entries are `nonlinear-regression-config`, `nonlinear-regression-artifact` and `nonlinear-regression-result`. Configurations include numeric X/Y, curve or validated expression, ordered starts/bounds, confidence level strictly between 0.5 and 1, and row/evaluation budgets. The method marker is `nonlinear_regression.least_squares.v1`, implementation 1.1.0.

Prediction accepts a strict version-1 JSON artifact and the same-named numeric X without requiring Y or rerunning optimization. It validates configuration, estimates, usable covariance, residual variance, degrees of freedom, fit range and provenance. Unknown fields, invalid bounds and unusable covariance are rejected. Results preserve source-row IDs and eligible/missing/excluded status with original-scale residuals.

2.9.10 Deming regression

`analysis.deming_regression` exports `DemingConfig`, `DemingResult` and `run_deming`. Schemas are `deming-regression-config` and `deming-regression-result`. The roles are `x_column` and `y_column`; `lambda_ratio` is positive finite Y-error variance divided by X-error variance. `comparable_units` enables identity-line interpretation.

`interval_method` is `jackknife_t`, `percentile_bootstrap` or `none`. Other fields are `confidence_level`, `bootstrap_resamples` and `seed`. Confidence is strictly between 0 and 1; seeds span 0–4,294,967,295. Bootstrap resamples are allowed only for percentile intervals. Omitted

interval-method settings preserve the historical interpretation: no interval without resamples, percentile intervals with them. Implementation 1.1.0 introduced the explicit choice; 1.0.0 artifacts keep their original meaning. Jackknife uncertainty occupies its own result fields, never bootstrap fields. Exclusions and cancellation are explicit and outputs are finite strict JSON.

2.9.11 Multiple imputation

`tabelz.analysis.multiple_imputation.compute_multiple_imputation` returns the pooled study. `complete_imputed_dataset` regenerates a selected completion. `MultipleImputationConfig` contains `variables`, `analysis`, `imputations`, `sweeps`, `seed`, `confidence_level` and `max_rows`. The analysis contains `model` (`ols` or `logistic`), `outcome`, `predictors` and logistic `event_value`. The method identity is `multiple_imputation.mice_rubin`.

The `tabelz.mi_completed_table` operation binds the source table, settings, imputation number, exclusions and completed-dataset fingerprint. Replay regenerates the selected chain and rejects a fingerprint mismatch. Observed cells are preserved; compatible source types, including ordered categorical levels, are retained. Otherwise numeric data widen to Float64 and categories to text with disclosure.

2.9.12 Classification trees

`tabelz.analysis.classification_tree.compute_classification_tree` takes a lazy or eager frame, frozen `ClassificationTreeConfig`, exclusions and optional `SplitAssignment`. Retained assignments must cover exactly the complete population with disjoint portions and every class in both. Stratification uses the discriminant-analysis split/class contract. Result schemas are `classification-tree-config`, `classification-tree-result` and `classification-tree-artifact`. The result retains row membership, probabilities, metrics, encoding and validated numeric node arrays. Fitting uses scikit-learn `DecisionTreeClassifier`, `gini` and `best`.

`tabelz.classification_tree_scores`, implementation 1.0.0, retains output name, target table/revision/exclusions, target string row IDs and immutable tree/encoding metadata. It scores without refitting, preserves target identities, leaves unusable rows null and rejects conflicting columns. Source-table identities are never substituted for target identities. The artifact is finite plain JSON, not a pickled estimator. Saved reports retain canonical settings, input/schema fingerprints, train/test membership, class order, warnings and implementation provenance.

2.9.13 Grouped summaries and cross-tabulation

`tabelz.analysis.tabulation.compute_tabulation` accepts an explicit bounded frame and frozen `TabulationConfig`, with source row identities and exclusions. `TabulationResult` separates source accounting, grouped numeric summaries, categorical counts, margins and percentage denominators. Integer frequencies represent replicated observations, not precision or survey weights. Numeric missingness is counted separately for the selected response; undefined small-sample uncertainty and zero-margin percentages remain explicit. The method uses deterministic category order, type-7 replicated quartiles and a sample-SD t interval for the mean at the retained confidence level. Unobserved category pairs are labelled observed zeros; structural zeros are never inferred. The desktop bounds its display and report cards while the full computed table remains available through CSV export. Budget violations fail explicitly rather than silently changing the calculation population. The `tabulation-config` and `tabulation-result` schema snapshots define the public serialization.

2.9.14 Distribution diagnostics

`tabelz.analysis.distribution_diagnostics.compute_distribution_diagnostics` accepts a frame and frozen `DistributionDiagnosticsConfig`. The `DistributionDiagnosticsResult` records the chosen diagnostic, alpha, finite complete-case population, exclusions, numerical statistic, decision, limits and bounded plot data. Shapiro–Wilk uses its supported sample-size range; normal Anderson–Darling accounts for estimated mean and variance. Two-sided single-outlier Grubbs inference requires normal observations and returns the candidate source-row identity without changing the source table. Fitted-Poisson goodness of fit validates integer counts, retains the infinite tail, discloses expected-count pooling and subtracts the fitted rate from its degrees of freedom. Degenerate normality/outlier inputs, a zero fitted Poisson rate and insufficient pooled degrees of freedom fail explicitly. The `distribution-diagnostics-config` and `distribution-diagnostics-result` snapshots define serialization.

2.9.15 Z, variance, and Poisson-rate inference

`tabelz.analysis.additional_basic_tests.compute_additional_basic_tests` uses frozen `AdditionalBasicTestsConfig` and returns `AdditionalBasicTestsResult`. Method-specific raw and entered-summary requests distinguish a known-sigma mean Z test, normal-population variance chi-square inference, exact one-rate Poisson inference and exact conditional two-rate comparison. Known sigma is never estimated from the submitted sample; variance and standard deviation keep distinct units. Events are nonnegative integers and exposures are finite and positive. Each event count and the aggregate count, combined across both groups when applicable, is bounded at $2^{53} - 1$; oversized inputs fail before lossy float conversion. Exact count intervals preserve nonzero upper limits for zero events. Conditional comparisons retain exposure units, the null rate ratio, the two-sided p-value convention and interval convention. Unbounded endpoints are represented explicitly, never JSON infinity. Entered summaries are source-independent; their saved reports retain the entered settings without inventing a dataset or row population. Raw requests retain the actual complete cases and source context. The `additional-basic-tests-config` and `additional-basic-tests-result` snapshots define public serialization.

The mean Z reference requires independent observations and a normal sample mean (or a justified large-sample approximation); variance inference requires an independent normal sample. Count-rate inference requires independent Poisson events and consistent positive exposure within each compared population. The flat configuration permits its neutral variance/unit defaults in serialized requests so standard JSON round trips remain valid. Nondefault incompatible choices, incompatible raw/summary roles, and unknown fields fail explicitly; the separate method request types reject irrelevant fields.

The existing report JSON shape is retained. A table snapshot with analysis key `additional_basic_tests` may omit dataset/revision only when its source context, settings, and result explicitly declare entered-summary input. It must retain zero source/excluded rows, no row membership or population fingerprint, and no table identity in its source context. The prospective precision and extended-power snapshots described below are another narrowly validated dataset-free case. Remaining analysis snapshots require dataset/revision. Existing projects remain readable; older application versions may reject these new summary snapshots on downgrade.

2.9.16 Friedman and runs tests

`tabelz.analysis.blocked_rank_tests.compute_blocked_rank_tests` accepts a bounded frame and frozen `BlockedRankTestsConfig`, with explicit row identities, exclusions and cancellation. `BlockedRankTestsResult` records source accounting, retained identities, method convention and

numerical outputs. Friedman requires an unreplicated complete block/treatment design; duplicate cells, incomplete designs and an unestimable all-tied design fail explicitly. The asymptotic chi-square statistic includes within-block tie correction, and rank summaries retain treatment labels. Runs inference uses the declared source order or an explicitly selected order column; response sorting is never a preparation step. Numeric values equal to the chosen reference are omitted and counted. Categorical inference requires exactly two observed categories. Exact and asymptotic inference retain distinct conventions and limits. The `blocked-rank-tests-config` and `blocked-rank-tests-result` snapshots define public serialization.

2.9.17 Dunnett and Games–Howell comparisons

`tabelz.analysis.additional_comparisons.compute_additional_comparisons` uses frozen `AdditionalComparisonsConfig` and returns `AdditionalComparisonsResult`. Dunnett compares eligible treatment means against the explicitly retained control under an independent normal model with common variance. Its single-step adjusted p-values and simultaneous intervals use a retained numerical integration seed. Games–Howell compares every pair of eligible groups using pairwise Welch variances and Satterthwaite degrees of freedom, with studentized-range adjustment for the selected family. It does not substitute Tukey inference. Outputs name each contrast's direction, family, uncertainty convention and data population; display/report adapters are bounded. The `additional-comparisons-config` and `additional-comparisons-result` snapshots define serialization.

2.9.18 One-sample and paired equivalence

`tabelz.analysis.equivalence_extensions.compute_equivalence_extensions` uses frozen `EquivalenceExtensionsConfig` and returns `EquivalenceExtensionsResult`. One-sample inference compares a mean with an explicit target; paired inference uses within-row first-minus-second values. Paired log-ratio inference requires positive responses and ratio bounds, computes on the logarithmic scale and back-transforms estimates and intervals geometrically. Missing values use one common paired population, never independently sorted columns. Both one-sided tests are required; the matching confidence level is $1 - 2\alpha$, including 90 percent at $\alpha = 0.05$. Zero standard error and invalid designs fail explicitly. The existing independent Welch `EquivalenceConfig`, `EquivalenceResult`, computation, defaults, method identity and serialized settings remain compatible. The extension's `equivalence-extensions-config` and `equivalence-extensions-result` snapshots are separate additive contracts. Saved results retain their original settings and implementation provenance; refitting remains explicit.

2.9.19 Probability distributions and seeded random data

`tabelz.analysis.distribution_utilities.compute_probability_calculation` accepts a strict `ProbabilityCalculationConfig` and returns `ProbabilityCalculationResult`. The distribution is a discriminated union on `family`: `normal` (`mean`, `standard_deviation`), `t` (`degrees_of_freedom`), `chi_square` (`degrees_of_freedom`), `f` (`numerator_df`, `denominator_df`), `binomial` (integer `trials`, `probability`), `poisson` (`mean`), `exponential` (`scale`, equal to the mean) and `weibull` (`shape`, `scale`). Requests carry at most 200 `x_values`, at most 200 `probabilities` strictly inside $(0, 1)$, an optional `IntervalRequest` with nullable bounds and explicit endpoint inclusion, and `curve_points` from 21 to 512 (default 201). An empty request fails. Value rows separate `density` (continuous) from `mass` (discrete) and report `cdf`, survival `sf` and their base-10 logarithms, which remain finite after the probability underflows. Discrete quantiles are exact smallest-k integer bisections of the CDF (lower tail) and survival function (upper tail); SciPy `ppf`/ `isf` are not used for discrete families because they fail in extreme tails of large

laws. Unbounded support limits and undefined moments are `null`. The method identity is `tabelz.analysis.distribution_utilities` version 1.0.0 and the result records the SciPy version.

`tabelz.analysis.distribution_sampling.generate_random_data` accepts `RandomDataConfig` (distribution, `rows`, `column_names`, `seed` from 0 to $2^{63}-1$, and `algorithm`) and returns the frame and a strict `RandomDataSummary`. At most 20 columns and 1,000,000 cells are accepted. Algorithm `pcg64-inverse-cdf-v1` gives column j a PCG64 generator from `SeedSequence(seed,spawn_key=(j,))`, maps the top 52 bits k of each raw output to $u = (k+\frac{1}{2})/2^{52}$, and returns SciPy `ppf(u)` for continuous families or the smallest-k quantile from exact CDF/survival tables for discrete families. NumPy distribution samplers are not used because NumPy does not guarantee their streams. The summary retains the seed, algorithm description, NumPy and SciPy versions and a SHA-256 fingerprint of column names, dtypes and little-endian values; generation is cancellable between chunks.

`tabelz.operations.random_data` defines the root operation `tabelz.random_table` (schema and implementation 1.0.0) with no input tables and `random_seed` equal to the configured seed. Its `RandomTableOperationParameters` retain the output table identity and name, the configuration, library versions and output fingerprint. Replay regenerates the table with row identities `<outputtableid>:<ordinal>` and fails when the fingerprint differs. The desktop binds the root to an empty source tuple, adopts the table as a derived dataset tab, and restores it on redo. The additive public snapshots are `probability-calculation-config`, `probability-calculation-result`, `random-data-config`, `random-data-summary` and `random-table-operation-parameters`.

A table report snapshot with analysis key `probability_calculator` has no dataset or source revision. Its `source_context` must declare `source_kind parameters` and must not contain a table identity, data revision, membership fingerprint or population fingerprint. Existing report shapes are unchanged; older application versions may reject these new snapshots on downgrade.

2.9.20 Survey descriptives

`tabelz.analysis.survey_descriptives.compute_survey_descriptives` accepts a frame and frozen `SurveyDescriptivesConfig` with explicit response, positive weight, stratum and within-stratum PSU roles. It implements one-stage with-replacement Taylor linearization for a weighted mean or a numeric zero/one proportion. `SurveyDescriptivesResult` retains estimate, standard error, t interval, sample/design counts and design degrees of freedom (PSUs minus strata) separately. Original design PSUs remain represented when response complete cases omit their measurements. Malformed design roles, singleton strata, strata with no observed response and unsupported domain exclusions fail explicitly. Proportion intervals are not clipped to zero/one. The public `survey-descriptives-config` and `survey-descriptives-result` snapshots describe serialization. Finite population corrections, multistage designs, replicate weights, subpopulation inference and survey regression are outside this increment.

2.9.21 Item consistency and scores

The focused engine `tabelz.analysis.item_analysis. compute_item_analysis` accepts a lazy frame, a strict frozen `ItemAnalysisConfig`, stable string row IDs, explicit row exclusions and optional cancellation. `ItemAnalysisResult` preserves the covariance-based and standardized Cronbach alpha coefficients (negative values reported, never clipped), per-item adjusted summaries with corrected item-total correlations and leave-one-item-out alphas, the bounded inter-item correlation matrix, the complete-case row IDs and the source/excluded/missing count partition. Public `item-analysis-config` and `item-analysis-result` schemas are generated with the other contract snapshots. Reverse-coded items are reflected once as low + high $-\,x$ using

the declared shared scale bounds, which are validated against every observed reverse-item value. Complete finite cases are the analyzed population; no value is imputed. The engine makes no latent-validity claim: alpha measures internal consistency of the declared item set, and that limit is carried in every result and report.

`tabelz.item_analysis_scores` implementation 1.0.0 retains the output name, mean/sum aggregation, declared item design, reverse flags with shared scale bounds, and fit-population row IDs. It aggregates the adjusted items over the recorded rows without refitting and leaves omitted rows missing. Creation rejects existing columns. Replaying an already-created saved snapshot succeeds only when every retained score agrees with that fixed design; conflicting values fail explicitly. The desktop adds one operation and supports undo/redo of the score column. Saved reports preserve settings, method identity `item_analysis.cronbach_alpha`, warnings, bounded tables and the projected input artifact checksum, imported source identity, fit-population fingerprint and source revision; they never rerun an analysis.

2.10 Capability interfaces

`tabelz.analysis.johnson_transform` exposes `fit_johnson_system` and `johnson_transform_value` for explicit SU, SB and shifted-SL transformations. The capability engine also accepts Gamma and Logistic models. The version-1 `tabelz.johnson_transform` operation retains the supplied family and parameters; `validate_johnson_transform_spec` and `apply_johnson_transform` validate and apply them without estimating another model.

`tabelz.analysis.batch_capability` exposes `BatchCapabilityConfig`, `BatchCapabilityResult` and `compute_batch_capability`; the configuration schema is `batch-capability-config`. The random-intercept decomposition separates within-batch residual variance and total variance. The result partitions missing, excluded, dropped-small-batch and analyzed rows. Its declared uncertainty method is the conservative Gaussian simultaneous construction in the capability chapter; historical results retain their recorded method. Neither this calculation nor a fitted-model diagnostic alone establishes process qualification.

2.11 Measurement-system interfaces

2.11.1 Measurement study worksheets

`tabelz.measurements.study_worksheets.generate_study_worksheet` accepts a strict `StudyWorksheetConfig` (layout `crossed`, `nested` or `attribute_agreement`; part and operator counts or explicit labels; trials; seed 0 to $2^{63} - 1$; optional blinded part labels and per-part standard values) and returns the collection frame with a strict `StudyWorksheetSummary`. The contracts allocate no measured values: the measurement column (`Measurement` for gage layouts, `Value` for attribute agreement) is an empty `Float64` slot. Limits are 10,000 parts, 1,000 operators, 100 trials per cell and 1,000,000 rows; nested designs require at least 2 operators and 2 parts per operator and reject explicit part labels. The run order is a seeded shuffle within operator blocks (algorithm `numpy-pcg64-operator-block-shuffle-v1`, method identity `tabelz.measurements.study_worksheets` 1.0.0); identical designs and seeds reproduce the exact run order and the summary records a SHA-256 run-order fingerprint.

`tabelz.operations.study_worksheet` defines the root operation `tabelz.study_worksheet` (schema and implementation 1.0.0) with no input tables and `random_seed` equal to the configured seed. Its `StudyWorksheetOperationParameters` retain the output table identity and name, the configuration, method and algorithm identity, implementation version and the run-order fingerprint. Replay regenerates the sheet and fails visibly when the recorded fingerprint differs.

The desktop binds the root to an empty source tuple, adopts the sheet as a derived dataset tab that never replaces the active dataset, and restores it on redo. The additive public snapshots are `study-worksheet-config`, `study-worksheet-summary` and `study-worksheet-operation-parameters`.

2.11.2 Attribute gage response

`tabelz.analysis.attribute_gage_response.compute_attribute_gage_response` accepts an explicit frame, frozen `AttributeGageResponseConfig`, stable row identities, exclusions and cancellation. Contract models live in `analysis.attribute_gage_response_contracts` and are re-exported by the engine module. The configuration names a reference-value, pass-count and trials column, the modeled event (`pass` or `fail`), a required finite threshold without a default, lower and upper transition probabilities (defaults 0.05 and 0.95) and a confidence level (default 0.95). Complete rows must hold whole-number counts with $1 \le n$ and $0 \le y \le n$; other complete values fail with the row named, and rows with a missing value are skipped and counted. Rows are pooled by exact reference value into at least three levels, and all-pass, all-fail or (quasi-)separated outcomes fail before fitting because no finite maximum-likelihood estimate exists. The grouped binomial probit model $\Phi(\beta_0 + \beta_1 x)$ is fitted with statsmodels GLM (IRLS) on centered and scaled reference values; the covariance is the unscaled inverse expected information. Response points $x_p = (\Phi^{-1}(p) - \beta_0)/\beta_1$ carry Fieller intervals that have no bounded pair of limits (null limits) exactly when $\beta_1^2 \le t^2 \operatorname{Var}(\beta_1)$; `fieller_interval` exposes the ratio interval. Bias is $x_{0.5}$ minus the threshold, and the spread and probit scale $1/|\beta_1|$ invert the Wald interval for the slope. The Pearson chi-square over the pooled levels has $k - 2$ degrees of freedom and only produces a warning. `AttributeGageResponseResult` partitions source rows into excluded, missing and analyzed rows, stores at most 2,000 levels and a 121-point curve with a pointwise band, and records the analyzed-row fingerprint. The additive public snapshots are `attribute-gage-response-config` and `attribute-gage-response-result`; method identity `attribute_gage_response.probit`, implementation version 1.0.0. Report cards use analysis key `attribute_gage_response`: four table cards and one plot card carrying the settings, counts, disclosure and source context. Existing report shapes are unchanged.

2.12 Additional monitoring interfaces

2.12.1 Finite-window moving averages

`tabelz.analysis.spc_moving_average` exposes `SpcMovingAverageConfig`, `SpcMovingAverageResult` and `compute_spc_moving_average`. The Phase-I or supplied historical center and sigma are frozen before monitoring. Declared stages reset rolling windows; startup policy determines when limits are available. Width one is supported. Only the supported limit-breach rule is applied: overlapping averages do not inherit independent-point run-rule or ARL guarantees. The public configuration snapshot is `spc-moving-average-config`.

2.12.2 Short-run standardization

`analysis.spc_short_run` exposes `ShortRunConfig`, `ShortRunResult` and `run_short_run`. Each product's center and positive sigma are supplied, never inferred from monitoring data. Z limits are fixed at ± 3 around zero. Span-two moving ranges use known-normal constants $d_2 = 2/\sqrt{\pi}$ and $d_3 = \sqrt{2 - 4/\pi}$: center d_2, lower limit zero and upper limit $d_2 + 3d_3$. These are three-standard-deviation range limits, not an exact tail-probability or ARL guarantee. Source order defines the sequence; stage boundaries and missing/excluded gaps reset adjacency. The default includes cross-product standardized pairs, with an explicit saved option to reset on

product changes. Results retain original-unit measurements and previous-row links as well as standardized signals.

The engine accepts explicit excluded row IDs and cancellation checks, returns finite strict JSON results, and identifies its method and implementation version separately from the application version, schema and Git revision. Its public schema catalog entries are `spc-short-run-config` and `spc-short-run-result`.

The moving-average interface additionally accepts the sample-standard-deviation baseline estimator and full-window startup policy. With full-window startup, partial statistics are retained but their limits and breach flags remain unavailable for the first $w - 1$ positions. This differs from an early Shewhart-rule startup.

2.13 Multivariate monitoring interfaces

2.13.1 Subgroup Hotelling and generalized variance

`tabelz.analysis.spc_multivariate_subgroups.compute_hotelling_subgroups` accepts an explicit frame, frozen `HotellingSubgroupsConfig`, stable row identities, exclusions and cancellation. Rational subgroups share one label of a declared subgroup column and a declared fixed size $n \geq 2$; roles follow a reference indicator value and must be uniform within a subgroup. Oversized and mixed-role subgroups fail explicitly, while short or incomplete subgroups are dropped whole and counted. `HotellingSubgroupsBaseline` freezes the grand mean, the pooled within-subgroup covariance (mean of $n - 1$ covariances), the Phase-II subgroup-mean limit $p(m+1)(n-1)/(mn-m-p+1)\, F_{1-\alpha}(p,\, mn-m-p+1)$ and Alt's b_1, b_2 generalized-variance limits with $|\Sigma|$ estimated as the mean reference $|S_i|$ over b_1. The baseline contract refuses $n = 1$, so an individual-observation limit cannot be stored as a subgroup baseline. When $n \leq p$ the generalized-variance fields are absent and a reason is required. `HotellingSubgroupsResult` partitions source rows into excluded, unassigned, incomplete, reference and monitoring rows and links each monitored subgroup to its member row identities. The additive public snapshots are `hotelling-subgroups-config` and `hotelling-subgroups-result`; method identity `hotelling_t2.phase2_subgroups`, implementation version 1.0.0. Report cards use analysis key `hotelling_subgroups` and carry the settings, baseline fingerprint and source context. Existing report shapes are unchanged.

2.13.2 MEWMA

`tabelz.analysis.spc_mewma.compute_mewma` accepts an explicit frame, frozen `MewmaConfig`, stable row identities, exclusions and cancellation. Roles follow the individual Hotelling chart's reference indicator value. `MewmaBaseline` freezes the reference sample mean and the $m - 1$ sample covariance S with $m > p$; a singular S fails explicitly. Complete monitoring rows are taken in table order with $Z_0 = \bar{x}$, $Z_i = \lambda x_i + (1-\lambda) Z_{i-1}$ and the exact finite-time covariance $\lambda[1-(1-\lambda)^{2i}]/(2-\lambda)\, S$; an optional stage column restarts Z and i on every label change, rows with missing values or blank stages are skipped without advancing i, and $\lambda = 1$ reproduces the individual Hotelling statistic. `covariance_factor` exposes the finite-time factor. The configuration requires a positive decision limit h without a default and records `limit_provenance` as `uncalibrated` (no source allowed) or `user_calibrated` (a non-empty `limit_source` of at most 500 characters). No field of the configuration, result or report carries an ARL or false-alarm probability. `MewmaResult` partitions source rows into excluded, incomplete, reference and monitoring rows and links each point to its row identity, stage, segment and step. The additive public snapshots are `mewma-config` and `mewma-result`; method identity `mewma.phase2`,

implementation version 1.0.0. Report cards use analysis key `mewma` and carry the settings, limit statement, baseline fingerprint and source context. Existing report shapes are unchanged.

2.14 Right-censored life distributions

`tabelz.analysis.life_distributions.fit_censored_life_distributions` returns `LifeDistResult` for right-censored exponential, Weibull, lognormal and loglogistic candidates fitted to one common population. Failures contribute density and right-censored records contribute survival. Results retain failed candidates, likelihood/AIC/BIC, parameter and life summaries, survival, hazard and comparison diagnostics. No free threshold is fitted.

`LifeFamiliesConfig`, `IntervalLifeConfig` and `AcceleratedLifeConfig` are typed desktop dataclasses; these reliability configurations and their numerical result dataclasses have no generated public Pydantic schema snapshots. Report blocks retain effective settings, numerical snapshots, source identity and exclusions, so saving and reopening preserves a result without refitting it.

2.15 Mixed-censored life distributions

`tabelz.analysis.life_censoring.fit_arbitrary_censored_life` returns `CensoredLifeResult` for the same families. Equal finite positive bounds are exact failures; right, left and strictly ordered interval encodings contribute survival, CDF and interval probability respectively. Missing endpoints have the desktop's explicit censoring semantics. Invalid bounds fail rather than being silently treated as exact observations. Every result retains censor-type counts.

2.16 Accelerated life models

`tabelz.analysis.accelerated_life.fit_accelerated_life` returns `AcceleratedLifeResult` for Weibull and lognormal right-censored AFT models. The user declares Arrhenius, inverse-power or linear stress relationships; Arrhenius converts Celsius to Kelvin and rejects nonpositive absolute temperature. Model families remain identified in use-stress predictions; failed candidates do not trigger an unlabeled substitute. Life and acceleration uncertainty uses fitted parameter covariance where estimable, with unavailable bounds and extrapolation disclosed. A common shape or dispersion across stresses is an assumption of these models.

`AltSurvivalCurve.interval_points` retains optional $(t, \text{lower}, \text{upper})$ pointwise survival bounds alongside the existing two-field mean points; its empty default preserves legacy construction. The pure `accelerated_life_intervals.pointwise_survival_interval` helper uses covariance ordered as $(\beta_0, \beta_1, \log \text{dispersion})$. Unsupported covariance yields unavailable bounds, preserving the mean curve. Survival intervals are conditional pointwise delta-method intervals, without simultaneous or family-selection coverage claims.

2.17 Precision planning

`tabelz.analysis.precision_planning.compute_precision_planning` takes frozen `PrecisionPlanningConfig` and returns `PrecisionPlanningResult`. Confidence, assumed spread/proportion, precision criterion and tolerance content have distinct meanings. The selected integer satisfies the requested criterion; its predecessor fails unless the declared lower bound binds. Normal tolerance factors and distribution-free order-statistic coverage retain the conventions of the existing tolerance interval engines. Planning works from parameters and does not construct a fictitious sample.

Public snapshots are `precision-planning-config` and `precision-planning-result`, and the parameter-only report table uses the analysis key `precision_planning`. A report retains the

complete typed settings and results, a planning source declaration, request identity and method provenance, and claims no dataset, revision or population. The report validator rejects incomplete or mismatched settings and results and fictitious source identity. Saving and reopening preserves immutable results; recalculation is explicit.

2.18 Prospective power planning

`tabelz.analysis.extended_power.compute_extended_power` takes frozen `ExtendedPowerConfig` and returns `ExtendedPowerResult`. Settings retain raw-unit assumptions, design, alternative, allocation, significance, target power, search bounds and curve settings. Rounded sample sizes are recalculated before publication. Discrete rate-power searches must account for nonmonotonic jumps rather than silently taking a smaller approximate solution. TOST true differences and equivalence margins remain separate parameters. No entered-parameter result constitutes evidence of observed post-hoc power. Cancellation and request generations suppress superseded publications.

Public snapshots are `extended-power-config` and `extended-power-result`. Parameter-only report tables use the analysis key `extended_power`, and a curve plot uses `extended_power_curve`. A report retains the complete typed settings and results, a planning source declaration, request identity and method provenance, and claims no dataset, revision or population. The additive report behavior keeps the existing report schema version, so older application versions may reject these snapshots on downgrade. Saving and reopening preserves immutable results; recalculation is explicit.

2.19 Stored regression scoring

`tabelz.analysis.model_scoring` owns strict `ModelScoringConfig`, `ModelScoringArtifact` and `ModelScoringResult` contracts. `fit_model_scoring` creates an artifact for OLS, WLS, binary logistic or Poisson regression. `score_model_scoring` applies a frozen artifact to a compatible target without fitting. Feature order, categorical references, transformations, coefficients, covariance and distribution/link identity belong to the artifact. Unknown categorical levels fail explicitly. Artifacts contain finite JSON values and are never pickled estimators.

OLS uncertainty distinguishes a mean response interval from an individual prediction interval, whose variance includes residual variance. Logistic and Poisson predictions use the response scale. Weight/exposure and unavailable uncertainty or influence conventions are disclosed. Eligible OLS fits expose leverage, studentized residuals, Cook's distance and DFBETAS, with bounded drawings and complete bounded row-aligned outputs. Stable row identity, missing/excluded membership and fit provenance remain distinct from target source identity. Saved report artifacts can be selected after reopening and used on another compatible table.

The `tabelz.operations.model_scoring.model_scoring_scores` builder creates a versioned score-column operation carrying safe artifact JSON and target row membership. The foundation registry validates and replays this operation; the desktop preserves lineage and undo/redo. Public schemas are `model-scoring-config`, `model-scoring-artifact`, `model-scoring-result` and `model-scoring-score-parameters`. Scoring parameter DTOs live in `tabelz.analysis.model_scoring_contracts`; explicit re-exports in `tabelz.operations.model_scoring` preserve their original class identities. The artifact and scoring entry point retain their canonical module import paths; the analysis facade additionally exports config, result and fit.

2.20 Stored-model profiling

`tabelz.analysis.stored_model_profiler` provides `ProfilerModelBinding`, `StoredProfilerConfig`, `StoredProfilerResult` and `compute_stored_model_profiler` for the desktop workflow. Bindings retain immutable `ModelScoringArtifact` objects; predictors and goals use the existing `PredictorSpecification` and `DesirabilityGoal` contracts. No numerical fitting or source table is required. Encoding, event identity, covariance and optional exposure come from the retained artifact. New scoring fits record complete/non-excluded numeric fitting ranges as optional provenance; older artifacts remain valid and cannot certify their missing ranges. The configuration bounds models to eight, predictors to sixteen, categorical combinations to 128, uniform candidates to 10,000 and local evaluations to 4096. All categorical combinations are enumerated. Numeric search and bounded refinement return a reproducible best found setting, not a global certification. OLS/WLS uncertainty uses retained covariance and residual t degrees of freedom; GLM uncertainty is approximate transformed link-scale Wald uncertainty. All intervals are pointwise mean-response intervals and not search-adjusted. The parameter-only spawned executor uses versioned JSON with a 64 MiB bound and a 180-second timeout. Its typed service snapshots settings, rejects mismatched replies and suppresses stale generations. Model-scenario report cards validate retained fit/config/result fingerprints and disallow a claim to the current dataset population. Saved/reopened settings replay frozen models.

The coefficient-only API below remains available for compatibility.

`analysis.model_optimization` exposes `ModelProfilerConfig`, `ModelSpecification`, `PredictorSpecification`, `DesirabilityGoal` and `compute_model_profiler`. This engine evaluates supplied coefficients; it does not refit a model. Numeric coefficients use the predictor name. Categorical indicators use an exact `predictor_level` or `predictor[level]` coefficient name and contribute only for that selected category. The linear predictor is returned directly for OLS, through the logistic inverse link for logistic models, or through the exponential inverse link for Poisson models.

Maximize and minimize goals interpolate linearly between the entered low and high response bounds and saturate in their respective preferred directions. Target goals have zero desirability at and beyond both bounds, with linear segments to a peak of one at an interior target. An omitted target uses the midpoint. Joint desirability is the weighted geometric mean, so any zero individual desirability makes it zero.

Numeric profiles contain 101 equally spaced settings; absent numeric bounds default to 0 and 10. Categorical profiles enumerate the entered categories. Other predictors remain at their current settings during each profile. Optimization samples 5000 numeric/categorical combinations with NumPy seed 42, then refines numeric settings with Nelder–Mead when positive desirability is available. It returns the best found setting, not a guarantee of the global optimum. Extrapolation warnings compare current numeric settings with supplied original fitting ranges.

Additional planning API limits. The extended-power engine permits sample-size bounds from 2 through 1000000, while the desktop defaults to 2–1000. Its optional seed is retained metadata; the power calculations are deterministic. Stored-profiler seed values support integers through $2^{63} - 1$ in the typed configuration; the desktop restricts them to $2^{53} - 1$ for exact numeric transport.

CHAPTER 3

Callable reference

This catalog records the callable signatures of the public analysis, operation and recipe exports, the direct numerical interfaces discussed in the reference, and the CSV/SQL importer. Class members are called on an instance unless shown as class methods. The following type catalog lists the fields accepted by settings objects and returned by typed results. A bare `*` separates positional arguments from keyword-only arguments. An argument without `=` is required. A union with `None` permits a missing value; it does not make an argument optional unless a default is shown. `pl` means Polars and `np` means NumPy. Probabilities use 0–1, except where a field explicitly declares another unit. The catalog is a syntax and structure reference. The preceding runnable examples demonstrate complete input construction and output access. Statistical assumptions and conditional constraints remain part of each method's contract; a type annotation alone does not establish that a dataset is suitable.

3.1 tabelz.analysis

3.1.1 tabelz.analysis.align_series

Align series with different observation dates without implicit value filling.

```
from tabelz.analysis import align_series
align_series(
    frames: tuple[pl.DataFrame, ...],
    time_columns: tuple[str, ...],
    *,
    join: JoinKind = 'outer',
) -> pl.DataFrame
```

3.1.2 tabelz.analysis.analysis_of_means.compute_analysis_of_means

Run one analysis of means on the included rows.

```
from tabelz.analysis.analysis_of_means import compute_analysis_of_means
compute_analysis_of_means(
    frame: pl.LazyFrame | pl.DataFrame,
    config: AnalysisOfMeansConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> AnalysisOfMeansResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `AnalysisOfMeansConfig`; fields and defaults are in Section 4.1.7. **Returns:** `AnalysisOfMeansResult`; field names and types are listed in Section 4.1.8.

3.1.3 tabelz.analysis.analyze_factorial

Fit a coded OLS factorial model and return bounded diagnostic payloads.

```
from tabelz.analysis import analyze_factorial
analyze_factorial(
    frame: pl.LazyFrame,
    response: str,
    factors: tuple[str, ...],
    *,
    include_two_factor_interactions: bool = True,
    confidence_level: float = 0.95,
) -> DoeAnalysisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

response
Name of the response column in the input table.

confidence_level
Confidence probability on the 0 to 1 scale, not a percentage.

Returns: `DoeAnalysisResult`; field names and types are listed in Section 4.1.137.

3.1.4 tabelz.analysis.analyze_time_series

Run one explicit diagnostic task without fitting or forecasting a model.

```
from tabelz.analysis import analyze_time_series
analyze_time_series(
    frame: pl.DataFrame,
    config: SeriesAnalysisConfig,
) -> SeriesAnalysisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `SeriesAnalysisConfig`; fields and defaults are in Section 4.1.632. **Returns:** `SeriesAnalysisResult`; field names and types are listed in Section 4.1.634.

3.1.5 tabelz.analysis.anova_post_hoc.compute_anova_post_hoc

```
from tabelz.analysis.anova_post_hoc import compute_anova_post_hoc
compute_anova_post_hoc(
    frame: pl.LazyFrame | pl.DataFrame,
    config: AnovaPostHocConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> AnovaPostHocResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `AnovaPostHocConfig`; fields and defaults are in Section 4.1.11.
Returns: `AnovaPostHocResult`; field names and types are listed in Section 4.1.12.

3.1.6 tabelz.analysis.assess_zivot_andrews

Run the Zivot-Andrews unit-root test with one endogenous structural break.

```
from tabelz.analysis import assess_zivot_andrews
assess_zivot_andrews(
    frame: pl.DataFrame,
    response_column: str,
    *,
    time_column: str | None = None,
    deterministic_term: Literal['constant', 'trend', 'both'] = 'both',
) -> ZivotAndrewsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

response_column
Name of the response column in the input table.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

Returns: `ZivotAndrewsResult`; field names and types are listed in Section 4.1.577.

3.1.7 tabelz.analysis.association_measures.compute_association

```
from tabelz.analysis.association_measures import compute_association
compute_association(
    frame: pl.LazyFrame | pl.DataFrame,
    config: AssociationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> AssociationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `AssociationConfig`; fields and defaults are in Section 4.1.14.
Returns: `AssociationResult`; field names and types are listed in Section 4.1.15.

3.1.8 tabelz.analysis.bayesian.conjugate.compute_bayes_binomial

```
from tabelz.analysis.bayesian.conjugate import compute_bayes_binomial
compute_bayes_binomial(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesBinomialConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesBinomialResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesBinomialConfig`; fields and defaults are in Section 4.1.25.
Returns: `BayesBinomialResult`; field names and types are listed in Section 4.1.26.

3.1.9 tabelz.analysis.bayesian.conjugate.compute_bayes_normal

```
from tabelz.analysis.bayesian.conjugate import compute_bayes_normal
compute_bayes_normal(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesNormalConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesNormalResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesNormalConfig`; fields and defaults are in Section 4.1.30.
Returns: `BayesNormalResult`; field names and types are listed in Section 4.1.31.

3.1.10 tabelz.analysis.bayesian.conjugate.compute_bayes_poisson

```
from tabelz.analysis.bayesian.conjugate import compute_bayes_poisson
compute_bayes_poisson(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesPoissonConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesPoissonResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesPoissonConfig`; fields and defaults are in Section 4.1.32. **Returns:** `BayesPoissonResult`; field names and types are listed in Section 4.1.33.

3.1.11 tabelz.analysis.bayesian.jzs.compute_bayes_correlation

```
from tabelz.analysis.bayesian.jzs import compute_bayes_correlation
compute_bayes_correlation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesCorrelationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesCorrelationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesCorrelationConfig`; fields and defaults are in Section 4.1.27. **Returns:** `BayesCorrelationResult`; field names and types are listed in Section 4.1.28.

3.1.12 tabelz.analysis.bayesian.jzs.compute_bayes_ttest

```
from tabelz.analysis.bayesian.jzs import compute_bayes_ttest
compute_bayes_ttest(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesTTestConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesTTestResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesTTestConfig`; fields and defaults are in Section 4.1.38.
Returns: `BayesTTestResult`; field names and types are listed in Section 4.1.39.

3.1.13 tabelz.analysis.bayesian.linear.compute_bayes_anova

```
from tabelz.analysis.bayesian.linear import compute_bayes_anova
compute_bayes_anova(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesAnovaConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesAnovaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesAnovaConfig`; fields and defaults are in Section 4.1.23.
Returns: `BayesAnovaResult`; field names and types are listed in Section 4.1.24.

3.1.14 tabelz.analysis.bayesian.linear.compute_bayes_regression

```
from tabelz.analysis.bayesian.linear import compute_bayes_regression
compute_bayes_regression(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BayesRegressionConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BayesRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BayesRegressionConfig`; fields and defaults are in Section 4.1.36. **Returns:** `BayesRegressionResult`; field names and types are listed in Section 4.1.37.

3.1.15 tabelz.analysis.begin_refresh

```
from tabelz.analysis import begin_refresh
begin_refresh(
    block: ResultTableBlock | PlotBlock | MethodNoteBlock,
) -> ResultTableBlock | PlotBlock | MethodNoteBlock
```

3.1.16 tabelz.analysis.begin_sheet_refresh

Explicitly begin a whole-sheet refresh for every non-text, non-frozen block.

```
from tabelz.analysis import begin_sheet_refresh
begin_sheet_refresh(
    sheet: AnalysisSheet,
) -> AnalysisSheet
```

Argument contract: `sheet` uses `AnalysisSheet`; fields and defaults are in Section 4.1.547. **Returns:** `AnalysisSheet`; field names and types are listed in Section 4.1.547.

3.1.17 tabelz.analysis.bootstrap_framework.compute_bootstrap_estimate

```
from tabelz.analysis.bootstrap_framework import compute_bootstrap_estimate
compute_bootstrap_estimate(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BootstrapEstimateConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BootstrapEstimateResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BootstrapEstimateConfig`; fields and defaults are in Section 4.1.44. **Returns:** `BootstrapEstimateResult`; field names and types are listed in Section 4.1.45.

3.1.18 tabelz.analysis.calendar_alignment_summary

Distinguish expected exchange closures from missing trading sessions.

```
from tabelz.analysis import calendar_alignment_summary
calendar_alignment_summary(
    frame: pl.DataFrame,
    time_column: str,
    spec: MarketCalendarSpec,
) -> dict[str, object]
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `MarketCalendarSpec`; fields and defaults are in Section 4.1.638.

3.1.19 tabelz.analysis.calendar_feature_frame

Create explicit session, weekday, month-end, and intervention regressors.

```
from tabelz.analysis import calendar_feature_frame
calendar_feature_frame(
    timestamps: tuple[date | datetime | str, ...],
    calendar: MarketCalendarSpec,
    *,
    interventions: dict[str, tuple[date | datetime | str, ...]] | None = None,
) -> pl.DataFrame
```

Argument contract: `calendar` uses `MarketCalendarSpec`; fields and defaults are in Section 4.1.638.

3.1.20 tabelz.analysis.canonical_correlation.compute_canonical_correlation

```
from tabelz.analysis.canonical_correlation import compute_canonical_correlation
compute_canonical_correlation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CanonicalConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
```

```
) -> CanonicalResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CanonicalConfig`; fields and defaults are in Section 4.1.46.
Returns: `CanonicalResult`; field names and types are listed in Section 4.1.49.

3.1.21 tabelz.analysis.capability_sixpack.compute_capability_comparison

The identical capability computation for each population, side by side.

```
from tabelz.analysis.capability_sixpack import compute_capability_comparison
compute_capability_comparison(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ComparisonConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ComparisonResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ComparisonConfig`; fields and defaults are in Section 4.1.53.
Returns: `ComparisonResult`; field names and types are listed in Section 4.1.54.

3.1.22 tabelz.analysis.capability_sixpack.compute_capability_multiple

One existing capability run per column against its own specification.

```
from tabelz.analysis.capability_sixpack import compute_capability_multiple
compute_capability_multiple(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MultipleConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MultipleResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MultipleConfig`; fields and defaults are in Section 4.1.55.
Returns: `MultipleResult`; field names and types are listed in Section 4.1.56.

3.1.23 tabelz.analysis.capability_sixpack.compute_capability_sixpack

```
from tabelz.analysis.capability_sixpack import compute_capability_sixpack
compute_capability_sixpack(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CapabilitySetup,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SixpackResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CapabilitySetup`; fields and defaults are in Section 4.1.51.
Returns: `SixpackResult`; field names and types are listed in Section 4.1.58.

3.1.24 tabelz.analysis.categorical_pca.compute_categorical_pca

```
from tabelz.analysis.categorical_pca import compute_categorical_pca
compute_categorical_pca(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CatpcaConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    tolerance: float = 1e-10,
    quantification_tolerance: float = 1e-08,
) -> CatpcaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CatpcaConfig`; fields and defaults are in Section 4.1.61.
Returns: `CatpcaResult`; field names and types are listed in Section 4.1.64.

3.1.25 tabelz.analysis.chaid.compute_chaid

```
from tabelz.analysis.chaid import compute_chaid
compute_chaid(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ChaidConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ChaidResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ChaidConfig`; fields and defaults are in Section 4.1.67. **Returns:** `ChaidResult`; field names and types are listed in Section 4.1.69.

3.1.26 tabelz.analysis.clustering.compute_kmeans_clustering

Compute bounded, deterministic k-means over complete finite cases.

```
from tabelz.analysis.clustering import compute_kmeans_clustering
compute_kmeans_clustering(
    frame: pl.LazyFrame,
    config: KMeansConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> KMeansResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `KMeansConfig`; fields and defaults are in Section 4.1.78. **Returns:** `KMeansResult`; field names and types are listed in Section 4.1.79.

3.1.27 tabelz.analysis.compare_analysis_goldens

Apply exact same-environment or tolerant cross-environment policy.

```
from tabelz.analysis import compare_analysis_goldens
compare_analysis_goldens(
    expected: AnalysisGolden,
    actual: AnalysisGolden,
    *,
```

```
    same_environment: bool,
) -> None
```

Argument contract: `expected` uses `AnalysisGolden`; fields and defaults are in Section 4.1.210.
Argument contract: `actual` uses `AnalysisGolden`; fields and defaults are in Section 4.1.210.

3.1.28 tabelz.analysis.complete_refresh

```
from tabelz.analysis import complete_refresh
complete_refresh(
    block: ResultTableBlock | PlotBlock | MethodNoteBlock,
    artifact_result: ResultRef | None = None,
) -> ResultTableBlock | PlotBlock | MethodNoteBlock
```

3.1.29 tabelz.analysis.component_fingerprint

Hash the explicit component projection, excluding display fields.

```
from tabelz.analysis import component_fingerprint
component_fingerprint(
    component: ResultComponent,
) -> str
```

3.1.30 tabelz.analysis.component_fingerprint_projection

Project one component onto its explicit fingerprint whitelist.

```
from tabelz.analysis import component_fingerprint_projection
component_fingerprint_projection(
    component: ResultComponent,
) -> ComponentFingerprint
```

3.1.31 tabelz.analysis.compute_additional_basic_tests

Compute one selected exact inference method.

```
from tabelz.analysis import compute_additional_basic_tests
compute_additional_basic_tests(
    frame: pl.DataFrame | pl.LazyFrame,
    config: AdditionalBasicTestsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> AdditionalBasicTestsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `AdditionalBasicTestsConfig`; fields and defaults are in Section 4.1.1. **Returns:** `AdditionalBasicTestsResult`; field names and types are listed in Section 4.1.2.

3.1.32 tabelz.analysis.compute_additional_comparisons

Compute one explicit multiple-comparison family.

```
from tabelz.analysis import compute_additional_comparisons
compute_additional_comparisons(
    frame: pl.DataFrame | pl.LazyFrame,
    config: AdditionalComparisonsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> AdditionalComparisonsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `AdditionalComparisonsConfig`; fields and defaults are in Section 4.1.3. **Returns:** `AdditionalComparisonsResult`; field names and types are listed in Section 4.1.4.

3.1.33 tabelz.analysis.compute_attribute_agreement

Calculate repeatability, reproducibility, and optional standard agreement.

```
from tabelz.analysis import compute_attribute_agreement
compute_attribute_agreement(
    frame: pl.DataFrame | pl.LazyFrame,
    config: AttributeAgreementConfig,
) -> AttributeAgreementResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `AttributeAgreementConfig`; fields and defaults are in Section 4.1.21. **Returns:** `AttributeAgreementResult`; field names and types are listed in Section 4.1.22.

3.1.34 tabelz.analysis.compute_blocked_rank_tests

Run one complete-block Friedman test or one ordered runs test.

```
from tabelz.analysis import compute_blocked_rank_tests
compute_blocked_rank_tests(
    frame: pl.DataFrame | pl.LazyFrame,
    config: BlockedRankTestsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> BlockedRankTestsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BlockedRankTestsConfig`; fields and defaults are in Section 4.1.40. **Returns:** `BlockedRankTestsResult`; field names and types are listed in Section 4.1.41.

3.1.35 tabelz.analysis.compute_classification_tree

Fit one bounded sklearn CART using only training rows for tuning.

```
from tabelz.analysis import compute_classification_tree
compute_classification_tree(
    frame: pl.LazyFrame,
    config: ClassificationTreeConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ClassificationTreeResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ClassificationTreeConfig`; fields and defaults are in Section 4.1.72. **Returns:** `ClassificationTreeResult`; field names and types are listed in Section 4.1.74.

3.1.36 tabelz.analysis.compute_cox_regression

Fit a fixed-covariate Cox model with Efron ties.

```
from tabelz.analysis import compute_cox_regression
compute_cox_regression(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CoxRegressionConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    row_id_column: str = '__tabelz_row_id',
) -> CoxRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

Argument contract: `config` uses `CoxRegressionConfig`; fields and defaults are in Section 4.1.91. **Returns:** `CoxRegressionResult`; field names and types are listed in Section 4.1.93.

3.1.37 tabelz.analysis.compute_cross_correlation

Run one bounded, deterministic two-series cross-correlation.

```
from tabelz.analysis import compute_cross_correlation
compute_cross_correlation(
    frame: pl.DataFrame | pl.LazyFrame,
    config: CrossCorrelationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    progress: Callable[[int, int, str], None] | None = None,
) -> CrossCorrelationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CrossCorrelationConfig`; fields and defaults are in Section 4.1.105. **Returns:** `CrossCorrelationResult`; field names and types are listed in Section 4.1.107.

3.1.38 tabelz.analysis.compute_distribution_diagnostics

Compute one bounded diagnostic, preserving source row identities.

```
from tabelz.analysis import compute_distribution_diagnostics
compute_distribution_diagnostics(
    frame: pl.DataFrame | pl.LazyFrame,
    config: DistributionDiagnosticsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> DistributionDiagnosticsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `DistributionDiagnosticsConfig`; fields and defaults are in Section 4.1.113. **Returns:** `DistributionDiagnosticsResult`; field names and types are listed in Section 4.1.114.

3.1.39 tabelz.analysis.compute_equivalence

Compute a bounded independent two-sample Welch TOST.

```
from tabelz.analysis import compute_equivalence
compute_equivalence(
    frame: pl.LazyFrame,
    config: EquivalenceConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    row_id_column: str = '__tabelz_row_id',
    cancel: Callable[[], bool] | None = None,
) -> EquivalenceResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `EquivalenceConfig`; fields and defaults are in Section 4.1.160. **Returns:** `EquivalenceResult`; field names and types are listed in Section 4.1.161.

3.1.40 tabelz.analysis.compute_equivalence_extensions

Compute one-sample or paired TOST without independently reordering rows.

```
from tabelz.analysis import compute_equivalence_extensions
compute_equivalence_extensions(
    frame: pl.LazyFrame,
    config: EquivalenceExtensionsConfig,
```

```
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> EquivalenceExtensionsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `EquivalenceExtensionsConfig`; fields and defaults are in Section 4.1.162. **Returns:** `EquivalenceExtensionsResult`; field names and types are listed in Section 4.1.163.

3.1.41 tabelz.analysis.compute_exploration_graph

Compute one complete bounded exploration graph over the retained rows.

```
from tabelz.analysis import compute_exploration_graph
compute_exploration_graph(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ExplorationGraphConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ExplorationGraphResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ExplorationGraphConfig`; fields and defaults are in Section 4.1.171. **Returns:** `ExplorationGraphResult`; field names and types are listed in Section 4.1.172.

3.1.42 tabelz.analysis.compute_extended_power

Compute power, an upward-rounded n/effect solve, or a bounded curve.

```
from tabelz.analysis import compute_extended_power
compute_extended_power(
    config: ExtendedPowerConfig,
    *,
    cancel: Callable[[], bool] | None = None,
) -> ExtendedPowerResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ExtendedPowerConfig`; fields and defaults are in Section 4.1.194.
Returns: `ExtendedPowerResult`; field names and types are listed in Section 4.1.195.

3.1.43 tabelz.analysis.compute_general_linear

```
from tabelz.analysis import compute_general_linear
compute_general_linear(
    frame: pl.DataFrame | pl.LazyFrame,
    config: GeneralLinearConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> GeneralLinearResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `GeneralLinearConfig`; fields and defaults are in Section 4.1.258.
Returns: `GeneralLinearResult`; field names and types are listed in Section 4.1.255.

3.1.44 tabelz.analysis.compute_linear_effects

Fit the configured complete-case additive model and infer factor effects.

```
from tabelz.analysis import compute_linear_effects
compute_linear_effects(
    frame: pl.DataFrame | pl.LazyFrame,
    config: LinearEffectsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Any | None = None,
) -> LinearEffectsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LinearEffectsConfig`; fields and defaults are in Section 4.1.247.
Returns: `LinearEffectsResult`; field names and types are listed in Section 4.1.250.

3.1.45 tabelz.analysis.compute_manova

```
from tabelz.analysis import compute_manova
compute_manova(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ManovaConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ManovaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ManovaConfig`; fields and defaults are in Section 4.1.275.
Returns: `ManovaResult`; field names and types are listed in Section 4.1.274.

3.1.46 tabelz.analysis.compute_mixed_model

Fit one additive Gaussian random-intercept model with explicit diagnostics.

```
from tabelz.analysis import compute_mixed_model
compute_mixed_model(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MixedModelConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    row_id_column: str = '__tabelz_row_id',
) -> MixedModelResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

Argument contract: `config` uses `MixedModelConfig`; fields and defaults are in Section 4.1.314.
Returns: `MixedModelResult`; field names and types are listed in Section 4.1.317.

3.1.47 tabelz.analysis.compute_mixed_model_structures

Fit exactly the declared F11 model and return bounded row-keyed diagnostics.

```
from tabelz.analysis import compute_mixed_model_structures
compute_mixed_model_structures(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MixedModelStructuresConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    row_id_column: str = '__tabelz_row_id',
) -> MixedModelStructuresResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

Argument contract: `config` uses `MixedModelStructuresConfig`; fields and defaults are in Section 4.1.310. **Returns:** `MixedModelStructuresResult`; field names and types are listed in Section 4.1.311.

3.1.48 tabelz.analysis.compute_panel_fixed_effects

Fit a one-way entity fixed-effects regression.

```
from tabelz.analysis import compute_panel_fixed_effects
compute_panel_fixed_effects(
    frame: pl.DataFrame | pl.LazyFrame,
    config: PanelFixedEffectsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    row_id_column: str = '__tabelz_row_id',
) -> PanelFixedEffectsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

Argument contract: `config` uses `PanelFixedEffectsConfig`; fields and defaults are in Section 4.1.377. **Returns:** `PanelFixedEffectsResult`; field names and types are listed in Section 4.1.379.

3.1.49 tabelz.analysis.compute_pareto

Aggregate causes in descending frequency with cumulative percentages.

```
from tabelz.analysis import compute_pareto
compute_pareto(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ParetoConfig,
) -> ParetoResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `ParetoConfig`; fields and defaults are in Section 4.1.380. **Returns:** `ParetoResult`; field names and types are listed in Section 4.1.381.

3.1.50 tabelz.analysis.compute_permutation_tests

Run one bounded, deterministic permutation test.

```
from tabelz.analysis import compute_permutation_tests
compute_permutation_tests(
    frame: pl.DataFrame | pl.LazyFrame,
    config: PermutationTestsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    progress: Callable[[int, int, str], None] | None = None,
) -> PermutationTestsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `PermutationTestsConfig`; fields and defaults are in Section 4.1.384. **Returns:** `PermutationTestsResult`; field names and types are listed in Section 4.1.385.

3.1.51 tabelz.analysis.compute_precision_planning

Return the smallest bounded integer n meeting the declared criterion.

```
from tabelz.analysis import compute_precision_planning
compute_precision_planning(
    config: PrecisionPlanningConfig,
    *,
    cancel: Callable[[], bool] | None = None,
) -> PrecisionPlanningResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `PrecisionPlanningConfig`; fields and defaults are in Section 4.1.395. **Returns:** `PrecisionPlanningResult`; field names and types are listed in Section 4.1.396.

3.1.52 tabelz.analysis.compute_probability_calculation

Evaluate density/mass, tails, quantiles and an interval for one distribution.

```
from tabelz.analysis import compute_probability_calculation
compute_probability_calculation(
    config: ProbabilityCalculationConfig,
) -> ProbabilityCalculationResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `ProbabilityCalculationConfig`; fields and defaults are in Section 4.1.128. **Returns:** `ProbabilityCalculationResult`; field names and types are listed in Section 4.1.129.

3.1.53 tabelz.analysis.compute_product_stability

Perform stability analysis and solve for product shelf-life.

```
from tabelz.analysis import compute_product_stability
compute_product_stability(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ProductStabilityConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ProductStabilityResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ProductStabilityConfig`; fields and defaults are in Section 4.1.409. **Returns:** `ProductStabilityResult`; field names and types are listed in Section 4.1.412.

3.1.54 tabelz.analysis.compute_regression_tree

Fit and validate a small tree with an untouched final test set.

```
from tabelz.analysis import compute_regression_tree
compute_regression_tree(
    frame: pl.LazyFrame,
    config: RegressionTreeConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> RegressionTreeResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RegressionTreeConfig`; fields and defaults are in Section 4.1.455. **Returns:** `RegressionTreeResult`; field names and types are listed in Section 4.1.457.

3.1.55 tabelz.analysis.compute_survey_descriptives

Compute the specified one stage with replacement Taylor ratio estimate.

```
from tabelz.analysis import compute_survey_descriptives
compute_survey_descriptives(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyDescriptivesConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyDescriptivesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyDescriptivesConfig`; fields and defaults are in Section 4.1.592. **Returns:** `SurveyDescriptivesResult`; field names and types are listed in Section 4.1.593.

3.1.56 tabelz.analysis.compute_tabulation

Compute a complete bounded tabulation; display truncation is a UI concern.

```
from tabelz.analysis import compute_tabulation
compute_tabulation(
    frame: pl.DataFrame | pl.LazyFrame,
    config: TabulationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> TabulationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `TabulationConfig`; fields and defaults are in Section 4.1.618. **Returns:** `TabulationResult`; field names and types are listed in Section 4.1.620.

3.1.57 tabelz.analysis.config_for_candidate

Return a manual configuration for a selected automatic candidate.

```
from tabelz.analysis import config_for_candidate
config_for_candidate(
    config: ForecastConfig,
    candidate: ForecastCandidate,
) -> ForecastConfig
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `ForecastConfig`; fields and defaults are in Section 4.1.626. **Argument contract:** `candidate` uses `ForecastCandidate`; fields and defaults are in Section 4.1.625. **Returns:** `ForecastConfig`; field names and types are listed in Section 4.1.626.

3.1.58 tabelz.analysis.convert_frequency

Apply an explicit, auditable frequency conversion to numeric columns.

```
from tabelz.analysis import convert_frequency
convert_frequency(
    frame: pl.DataFrame,
    time_column: str,
    rule: FrequencyRule,
    aggregation: AggregationKind,
    timezone: str = 'UTC',
) -> pl.DataFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

3.1.59 tabelz.analysis.correlation_extensions.compute_kendall

```
from tabelz.analysis.correlation_extensions import compute_kendall
compute_kendall(
    frame: pl.LazyFrame | pl.DataFrame,
    config: KendallConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
```

```
    cancel: Callable[[], bool] | None = None,
) -> KendallResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `KendallConfig`; fields and defaults are in Section 4.1.82. **Returns:** `KendallResult`; field names and types are listed in Section 4.1.84.

3.1.60 tabelz.analysis.correlation_extensions.compute_partial_correlation

```
from tabelz.analysis.correlation_extensions import compute_partial_correlation
compute_partial_correlation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: PartialCorrelationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> PartialCorrelationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `PartialCorrelationConfig`; fields and defaults are in Section 4.1.85. **Returns:** `PartialCorrelationResult`; field names and types are listed in Section 4.1.86.

3.1.61 tabelz.analysis.correspondence_analysis.run_correspondence

Compute bounded simple CA with deterministic category order and SVD signs.

```
from tabelz.analysis.correspondence_analysis import run_correspondence
run_correspondence(
    frame: pl.DataFrame | pl.LazyFrame,
    config: CorrespondenceConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CorrespondenceResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CorrespondenceConfig`; fields and defaults are in Section 4.1.88.
Returns: `CorrespondenceResult`; field names and types are listed in Section 4.1.89.

3.1.62 tabelz.analysis.cox_time_dependent.compute_cox_time_dependent

Fit the counting-process Cox model with the declared layout and time function.

```
from tabelz.analysis.cox_time_dependent import compute_cox_time_dependent
compute_cox_time_dependent(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CoxTimeDependentConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CoxTimeDependentResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CoxTimeDependentConfig`; fields and defaults are in Section 4.1.103. **Returns:** `CoxTimeDependentResult`; field names and types are listed in Section 4.1.104.

3.1.63 tabelz.analysis.create_analysis_golden

Create a self-verifying golden after resolving every referenced payload.

```
from tabelz.analysis import create_analysis_golden
create_analysis_golden(
    result: AnalysisResult,
    receipt: ProvenanceReceipt,
    resolver: PayloadResolver | None = None,
) -> AnalysisGolden
```

result
Typed result returned by the corresponding analysis.
receipt
Provenance receipt paired with the analysis result; None omits receipt rendering when supported.

Argument contract: `result` uses `AnalysisResult`; fields and defaults are in Section 4.1.518. **Argument contract:** `receipt` uses `ProvenanceReceipt`; fields and defaults are in Section 4.6.12. **Returns:** `AnalysisGolden`; field names and types are listed in Section 4.1.210.

3.1.64 tabelz.analysis.create_factorial_design

Create a run sheet with explicit standard/run order and an empty response.

```
from tabelz.analysis import create_factorial_design
create_factorial_design(
    config: FactorialDesignConfig,
) -> pl.DataFrame
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `FactorialDesignConfig`; fields and defaults are in Section 4.1.141.

3.1.65 tabelz.analysis.create_locked_profile_registry

Build a registry preloaded with the locked profile definitions.

```
from tabelz.analysis import create_locked_profile_registry
create_locked_profile_registry(
) -> ProfileRegistry
```

3.1.66 tabelz.analysis.create_profile_definition

Build a self-verifying profile definition with its computed fingerprint.

```
from tabelz.analysis import create_profile_definition
create_profile_definition(
    name: str,
    profile_version: str,
    settings: Iterable[ProfileSetting],
) -> ProfileDefinition
```

Returns: `ProfileDefinition`; field names and types are listed in Section 4.1.421.

3.1.67 tabelz.analysis.create_registered_run_profile

Build a registered run-profile snapshot requiring an exact registry match.

```
from tabelz.analysis import create_registered_run_profile
create_registered_run_profile(
    registry: ProfileRegistry,
    name: str,
    profile_version: str,
) -> RegisteredRunProfile
```

Returns: `RegisteredRunProfile`; field names and types are listed in Section 4.1.424.

3.1.68 tabelz.analysis.curve_estimation.compute_curve_estimation

Curve estimation at 1.1.0: a blank predictor defaults to input-table positions.

```
from tabelz.analysis.curve_estimation import compute_curve_estimation
compute_curve_estimation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CurveEstimationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CurveEstimationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CurveEstimationConfig`; fields and defaults are in Section 4.1.109. **Returns:** `CurveEstimationResult`; field names and types are listed in Section 4.1.110.

3.1.69 tabelz.analysis.data_fingerprint

Hash values, order, names, and dtypes for replay verification.

```
from tabelz.analysis import data_fingerprint
data_fingerprint(
    frame: pl.DataFrame,
    columns: tuple[str, ...] | None = None,
) -> str
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

3.1.70 tabelz.analysis.declared_profile_fingerprint

Hash the explicit declared-profile projection.

```
from tabelz.analysis import declared_profile_fingerprint
declared_profile_fingerprint(
    profile: RunProfile,
) -> str
```

3.1.71 tabelz.analysis.declared_profile_identity

Project a run profile onto its declared identity, without lineage.

```
from tabelz.analysis import declared_profile_identity
declared_profile_identity(
    profile: RunProfile,
) -> DeclaredProfileIdentity
```

3.1.72 tabelz.analysis.derive_custom_run_profile

Derive altered settings as a custom run profile plus receipt-only lineage.

```
from tabelz.analysis import derive_custom_run_profile
derive_custom_run_profile(
    base: RegisteredRunProfile,
    settings: Iterable[ProfileSetting],
) -> tuple[CustomRunProfile, ModifiedLineage]
```

Argument contract: `base` uses `RegisteredRunProfile`; fields and defaults are in Section 4.1.424.

3.1.73 tabelz.analysis.derive_time_series_features

Append explicitly named lag and rolling features while retaining source rows.

```
from tabelz.analysis import derive_time_series_features
derive_time_series_features(
    frame: pl.DataFrame,
    features: tuple[DerivedFeatureSpec, ...],
) -> pl.DataFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.1.74 tabelz.analysis.descriptives

Compute mean, median, sample standard deviation, and finite extremes.

```
from tabelz.analysis import descriptives
descriptives(
    values: Iterable[float | None],
    context: AnalysisContext,
    *,
    quantiles: Iterable[float] = (),
    weights: Sequence[float] | None = None,
) -> AnalysisOutcome
```

values
One-dimensional observations in their original measurement units; None represents missing input where accepted.

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

quantiles
Requested quantile probabilities between 0 and 1; the default empty tuple requests none.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.75 tabelz.analysis.difference_finder_options_from_payload

Strictly reconstruct options, including a persisted EntityKey mapping.

```
from tabelz.analysis import difference_finder_options_from_payload
difference_finder_options_from_payload(
    payload: Mapping[str, object],
) -> DifferenceFinderOptions
```

Returns: `DifferenceFinderOptions`; field names and types are listed in Section 4.1.112.

3.1.76 tabelz.analysis.distribution_data

Return exact bounded distribution summaries and deterministic drawable samples.

```
from tabelz.analysis import distribution_data
distribution_data(
    frame: pl.LazyFrame,
    y: str,
    *,
    group: str | None = None,
    groups: tuple[str, ...] = (),
    group_filters: Mapping[str, tuple[str, ...]] | None = None,
    bin_count: int = 40,
    max_points: int = 5000,
```

```
    max_groups: int = 36,
    seed: int = 0,
) -> DistributionData
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

seed
Integer seed for repeatable randomized work.

Returns: `DistributionData`; field names and types are listed in Section 4.1.387.

3.1.77 tabelz.analysis.doe_power.compute_doe_power_table

The generic-workflow entry point.

```
from tabelz.analysis.doe_power import compute_doe_power_table
compute_doe_power_table(
    frame: object,
    config: DoePowerConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> DoePowerResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `DoePowerConfig`; fields and defaults are in Section 4.1.142.
Returns: `DoePowerResult`; field names and types are listed in Section 4.1.144.

3.1.78 tabelz.analysis.doe_taguchi_analysis.analyze_taguchi

Summaries, response tables, the additive prediction and optional ANOVA.

```
from tabelz.analysis.doe_taguchi_analysis import analyze_taguchi
analyze_taguchi(
    frame: pl.LazyFrame | pl.DataFrame,
    config: TaguchiAnalysisConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> TaguchiAnalysisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `TaguchiAnalysisConfig`; fields and defaults are in Section 4.1.154. **Returns:** `TaguchiAnalysisResult`; field names and types are listed in Section 4.1.155.

3.1.79 tabelz.analysis.doe_taguchi_design.compute_taguchi_design_table

The generic-workflow entry point: design planning uses no table rows.

```
from tabelz.analysis.doe_taguchi_design import compute_taguchi_design_table
compute_taguchi_design_table(
    frame: object,
    config: TaguchiDesignConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> TaguchiDesign
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `TaguchiDesignConfig`; fields and defaults are in Section 4.1.159. **Returns:** `TaguchiDesign`; field names and types are listed in Section 4.1.158.

3.1.80 tabelz.analysis.effects_table

```
from tabelz.analysis import effects_table
effects_table(
    result: DoeAnalysisResult,
) -> pl.DataFrame
```

result
Typed result returned by the corresponding analysis.

Argument contract: `result` uses `DoeAnalysisResult`; fields and defaults are in Section 4.1.137.

3.1.81 tabelz.analysis.envelope_fingerprint

Hash the explicit envelope projection, independent of representation.

```
from tabelz.analysis import envelope_fingerprint
envelope_fingerprint(
    result: AnalysisResult,
) -> str
```

result
Typed result returned by the corresponding analysis.

Argument contract: `result` uses `AnalysisResult`; fields and defaults are in Section 4.1.518.

3.1.82 tabelz.analysis.explain_selection

Explain an explicit selected population relative to an explicit comparison population.

```
from tabelz.analysis import explain_selection
explain_selection(
    selected: Mapping[str, Sequence[object]],
    comparison: Mapping[str, Sequence[object]],
    *,
    context: AnalysisContext,
    options: SelectionExplainerOptions | None = None,
    selected_population_id: str = 'selected',
    comparison_population_id: str = 'comparison',
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.83 tabelz.analysis.export_block

Export directly from typed contracts; unsupported combinations fail explicitly.

```
from tabelz.analysis import export_block
export_block(
    block: SheetBlock,
    format: SheetExportFormat,
    *,
    graph: OperationGraph | None = None,
```

```
    resolver: PayloadResolver | None = None,
    result: AnalysisResult | None = None,
) -> SheetExport
```

result
Typed result returned by the corresponding analysis.

Returns: `SheetExport`; field names and types are listed in Section 4.1.554.

3.1.84 tabelz.analysis.factor_analysis.compute_factor_analysis

Compute bounded, deterministic maximum-likelihood exploratory factors.

```
from tabelz.analysis.factor_analysis import compute_factor_analysis
compute_factor_analysis(
    frame: pl.LazyFrame,
    config: FaConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> FaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `FaConfig`; fields and defaults are in Section 4.1.199. **Returns:** `FaResult`; field names and types are listed in Section 4.1.200.

3.1.85 tabelz.analysis.fail_refresh

```
from tabelz.analysis import fail_refresh
fail_refresh(
    block: ResultTableBlock | PlotBlock | MethodNoteBlock,
    message: str,
) -> ResultTableBlock | PlotBlock | MethodNoteBlock
```

3.1.86 tabelz.analysis.find_acceptance_plan

Find the smallest single-sample plan meeting AQL/LTPD risk constraints.

```
from tabelz.analysis import find_acceptance_plan
find_acceptance_plan(
    aql: float,
    ltpd: float,
    *,
    producer_risk: float = 0.05,
    consumer_risk: float = 0.1,
    lot_size: int | None = None,
    max_sample_size: int = 5000,
) -> AcceptancePlan
```

Returns: `AcceptancePlan`; field names and types are listed in Section 4.1.278.

3.1.87 tabelz.analysis.find_differences

Compare two explicit, equally shaped tabular populations deterministically.

```
from tabelz.analysis import find_differences
find_differences(
    first: Mapping[str, Sequence[object]],
    second: Mapping[str, Sequence[object]],
    *,
    context: AnalysisContext,
    options: DifferenceFinderOptions | None = None,
    specifications: Mapping[str, Specification] | None = None,
    first_population_id: str = 'first',
    second_population_id: str = 'second',
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.88 tabelz.analysis.fit_garch

Fit a GARCH(p,q), defaulting to heavy-tailed standardized Student errors.

```
from tabelz.analysis import fit_garch
fit_garch(
    frame: pl.DataFrame,
    response_column: str,
    *,
    horizon: int = 12,
    p: int = 1,
    q: int = 1,
```

```
    distribution: Literal['normal', 't'] = 't',
    mean: Literal['constant', 'zero'] = 'constant',
    time_column: str | None = None,
    confidence_level: float = 0.95,
    calendar_name: str | None = None,
    timezone: str = 'UTC',
) -> VolatilityForecastResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

response_column
Name of the response column in the input table.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

confidence_level
Confidence probability on the 0 to 1 scale, not a percentage.

Returns: `VolatilityForecastResult`; field names and types are listed in Section 4.1.201.

3.1.89 tabelz.analysis.fit_model_scoring

Fit one supported regression family and retain its scoring artifact.

```
from tabelz.analysis import fit_model_scoring
fit_model_scoring(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ModelScoringConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Any | None = None,
) -> ModelScoringResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ModelScoringConfig`; fields and defaults are in Section 4.1.326.
Returns: `ModelScoringResult`; field names and types are listed in Section 4.1.329.

3.1.90 tabelz.analysis.fit_multivariate_system

Fit VAR or VECM with Johansen rank and predictive-association diagnostics.

```
from tabelz.analysis import fit_multivariate_system
fit_multivariate_system(
    frame: pl.DataFrame,
    columns: tuple[str, ...],
    *,
    horizon: int = 12,
    lag_order: int = 1,
    cointegration_rank: int | None = None,
    deterministic: Literal['none', 'constant', 'trend'] = 'constant',
    time_column: str | None = None,
    confidence_level: float = 0.95,
    simulation_draws: int = 500,
    random_seed: int = 0,
    units: tuple[str, ...] = (),
    calendar_name: str | None = None,
    timezone: str = 'UTC',
) -> MultivariateTimeSeriesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

confidence_level
Confidence probability on the 0 to 1 scale, not a percentage.

random_seed
Integer seed for repeatable randomized work.

Returns: `MultivariateTimeSeriesResult`; field names and types are listed in Section 4.1.662.

3.1.91 tabelz.analysis.fit_weibull_reliability

Fit a two-parameter Weibull model by MLE with right censoring.

```
from tabelz.analysis import fit_weibull_reliability
fit_weibull_reliability(
    times: list[float] | tuple[float, ...] | np.ndarray,
    failed: list[bool] | tuple[bool, ...] | np.ndarray,
    *,
    confidence_level: float = 0.95,
) -> ReliabilityResult
```

confidence_level
Confidence probability on the 0 to 1 scale, not a percentage.

Returns: `ReliabilityResult`; field names and types are listed in Section 4.1.279.

3.1.92 tabelz.analysis.forecast_dataset

Return a tidy dataset containing rolling, final, and future forecasts.

```
from tabelz.analysis import forecast_dataset
forecast_dataset(
    result: ForecastStudyResult,
) -> pl.DataFrame
```

result
Typed result returned by the corresponding analysis.

Argument contract: `result` uses `ForecastStudyResult`; fields and defaults are in Section 4.1.650.

3.1.93 tabelz.analysis.forecast_time_series

Fit, validate, and forecast one explicitly ordered regular series.

```
from tabelz.analysis import forecast_time_series
forecast_time_series(
    frame: pl.DataFrame,
    config: ForecastConfig,
) -> ForecastResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `ForecastConfig`; fields and defaults are in Section 4.1.626.
Returns: `ForecastResult`; field names and types are listed in Section 4.1.630.

3.1.94 tabelz.analysis.freeze_block

Replace a live block by an immutable, visibly labeled snapshot with its identity.

```
from tabelz.analysis import freeze_block
freeze_block(
    block: ResultTableBlock | PlotBlock | MethodNoteBlock,
) -> FrozenSnapshotBlock
```

Returns: `FrozenSnapshotBlock`; field names and types are listed in Section 4.1.549.

3.1.95 tabelz.analysis.generalized_linear_models.compute_generalized_linear

```
from tabelz.analysis.generalized_linear_models import compute_generalized_linear
compute_generalized_linear(
    frame: pl.LazyFrame | pl.DataFrame,
    config: GeneralizedLinearConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> GeneralizedLinearResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `GeneralizedLinearConfig`; fields and defaults are in Section 4.1.202. **Returns:** `GeneralizedLinearResult`; field names and types are listed in Section 4.1.203.

3.1.96 tabelz.analysis.generate_random_data

Generate one bounded table; identical configs produce identical bytes.

```
from tabelz.analysis import generate_random_data
generate_random_data(
    config: RandomDataConfig,
    *,
    cancelled: Callable[[], bool] | None = None,
    progress: Callable[[float], None] | None = None,
) -> RandomDataResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

cancelled
Zero-argument cancellation callback; return True to stop work.

Argument contract: `config` uses `RandomDataConfig`; fields and defaults are in Section 4.1.132. **Returns:** `RandomDataResult`; field names and types are listed in Section 4.1.115.

3.1.97 tabelz.analysis.guided_scoring_columns.compute_guided_scores

```
from tabelz.analysis.guided_scoring_columns import compute_guided_scores
compute_guided_scores(
    frame: pl.LazyFrame,
    config: GuidedScoringConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: object = None,
) -> GuidedScoreResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `GuidedScoringConfig`; fields and defaults are in Section 4.1.214. **Returns:** `GuidedScoreResult`; field names and types are listed in Section 4.1.211.

3.1.98 tabelz.analysis.hierarchical_clustering.compute_hierarchical_clustering

Compute bounded, deterministic agglomerative clustering.

```
from tabelz.analysis.hierarchical_clustering import compute_hierarchical_clustering
compute_hierarchical_clustering(
    frame: pl.LazyFrame,
    config: HierarchicalClusteringConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> HierarchicalClusteringResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `HierarchicalClusteringConfig`; fields and defaults are in Section 4.1.215. **Returns:** `HierarchicalClusteringResult`; field names and types are listed in Section 4.1.216.

3.1.99 tabelz.analysis.histogram_bin_selection

Resolve one rendered histogram bar to stable source row identities on demand.

```
from tabelz.analysis import histogram_bin_selection
histogram_bin_selection(
    frame: pl.LazyFrame,
    histogram: HistogramData,
    bin_index: int,
    *,
    table_namespace: str,
) -> Selection
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

Argument contract: `histogram` uses `HistogramData`; fields and defaults are in Section 4.1.392.
Returns: `Selection`; field names and types are listed in Section 4.8.2.

3.1.100 tabelz.analysis.histogram_data

Aggregate a numeric source into a fixed-size histogram payload.

```
from tabelz.analysis import histogram_data
histogram_data(
    frame: pl.LazyFrame,
    column: str,
    *,
    bin_count: int = 40,
) -> HistogramData
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

Returns: `HistogramData`; field names and types are listed in Section 4.1.392.

3.1.101 tabelz.analysis.item_analysis.compute_item_analysis

Compute bounded, deterministic item-consistency statistics.

```
from tabelz.analysis.item_analysis import compute_item_analysis
compute_item_analysis(
    frame: pl.LazyFrame,
    config: ItemAnalysisConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ItemAnalysisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ItemAnalysisConfig`; fields and defaults are in Section 4.1.220. **Returns:** `ItemAnalysisResult`; field names and types are listed in Section 4.1.221.

3.1.102 tabelz.analysis.life_distribution_analysis.compute_life_distribution

Fit one family with a declared threshold; add the nonparametric estimate.

```
from tabelz.analysis.life_distribution_analysis import compute_life_distribution
compute_life_distribution(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LifeDistributionConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> LifeDistributionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LifeDistributionConfig`; fields and defaults are in Section 4.1.225. **Returns:** `LifeDistributionResult`; field names and types are listed in Section 4.1.226.

3.1.103 tabelz.analysis.life_distribution_analysis.compute_nonparametric_life

Kaplan-Meier, or Turnbull when any record is left or interval censored.

```
from tabelz.analysis.life_distribution_analysis import compute_nonparametric_life
```

```
compute_nonparametric_life(
    frame: pl.LazyFrame | pl.DataFrame,
    config: NonparametricLifeConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> NonparametricLifeResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `NonparametricLifeConfig`; fields and defaults are in Section 4.1.230. **Returns:** `NonparametricLifeResult`; field names and types are listed in Section 4.1.231.

3.1.104 tabelz.analysis.life_distribution_id.compute_life_distribution_id

Probability plots, AD statistics, correlations and percentiles per family.

```
from tabelz.analysis.life_distribution_id import compute_life_distribution_id
compute_life_distribution_id(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LifeDistributionIdConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> LifeDistributionIdResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LifeDistributionIdConfig`; fields and defaults are in Section 4.1.233. **Returns:** `LifeDistributionIdResult`; field names and types are listed in Section 4.1.234.

3.1.105 tabelz.analysis.life_regression.compute_life_regression

Fit the location-scale regression and predict percentiles.

```
from tabelz.analysis.life_regression import compute_life_regression
compute_life_regression(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LifeRegressionConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> LifeRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LifeRegressionConfig`; fields and defaults are in Section 4.1.238. **Returns:** `LifeRegressionResult`; field names and types are listed in Section 4.1.239.

3.1.106 tabelz.analysis.life_tables.compute_life_table

Actuarial life table per stratum with the declared intervals (implementation 1.2.0; "legacy" replays 1.1.0, without the median and the grouped comparison).

```
from tabelz.analysis.life_tables import compute_life_table
compute_life_table(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LifeTableConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    legacy: bool = False,
) -> LifeTableResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LifeTableConfig`; fields and defaults are in Section 4.1.243.
Returns: `LifeTableResult`; field names and types are listed in Section 4.1.244.

3.1.107 tabelz.analysis.linear_fit_data

Fit an OLS line from sufficient statistics without materializing source rows.

```
from tabelz.analysis import linear_fit_data
linear_fit_data(
    frame: pl.LazyFrame,
    x: str,
    y: str,
) -> LinearFitData
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

Returns: `LinearFitData`; field names and types are listed in Section 4.1.393.

3.1.108 tabelz.analysis.loglinear.compute_loglinear

```
from tabelz.analysis.loglinear import compute_loglinear
compute_loglinear(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LoglinearConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> LoglinearResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LoglinearConfig`; fields and defaults are in Section 4.1.262. **Returns:** `LoglinearResult`; field names and types are listed in Section 4.1.266.

3.1.109 tabelz.analysis.loglinear_selection.compute_loglinear_selection

```
from tabelz.analysis.loglinear_selection import compute_loglinear_selection
compute_loglinear_selection(
    frame: pl.LazyFrame | pl.DataFrame,
    config: LoglinearSelectionConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> LoglinearSelectionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `LoglinearSelectionConfig`; fields and defaults are in Section 4.1.267. **Returns:** `LoglinearSelectionResult`; field names and types are listed in Section 4.1.268.

3.1.110 tabelz.analysis.mark_source_changed

Mark matching live blocks stale; this intentionally never invokes an analysis.

```
from tabelz.analysis import mark_source_changed
mark_source_changed(
    catalog: AnalysisSheetCatalog,
    changed_dependency_ids: Iterable[str],
) -> AnalysisSheetCatalog
```

catalog
TableCatalog receiving imported tables; close it after use to release importer-owned resources.

Argument contract: `catalog` uses `AnalysisSheetCatalog`; fields and defaults are in Section 4.1.548. **Returns:** `AnalysisSheetCatalog`; field names and types are listed in Section 4.1.548.

3.1.111 tabelz.analysis.market_schedule

Return valid sessions and exchange open/close timestamps in the requested timezone.

```
from tabelz.analysis import market_schedule
market_schedule(
    spec: MarketCalendarSpec,
    start: date | datetime,
    end: date | datetime,
) -> pl.DataFrame
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `MarketCalendarSpec`; fields and defaults are in Section 4.1.638.

3.1.112 tabelz.analysis.materialize_vintage

Materialize the latest revision actually available at an as-of date.

```
from tabelz.analysis import materialize_vintage
materialize_vintage(
    frame: pl.DataFrame,
    spec: InformationAvailabilitySpec,
    as_of: date | datetime | str,
) -> pl.DataFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `InformationAvailabilitySpec`; fields and defaults are in Section 4.1.637.

3.1.113 tabelz.analysis.mean_confidence_interval

Compute the normal confidence interval for the mean of finite observations.

```
from tabelz.analysis import mean_confidence_interval
mean_confidence_interval(
    values: Iterable[float | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

values
One-dimensional observations in their original measurement units; None represents missing input where accepted.

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.114 tabelz.analysis.meta_analysis.compute_meta_analysis

```
from tabelz.analysis.meta_analysis import compute_meta_analysis
compute_meta_analysis(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MetaAnalysisConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MetaAnalysisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MetaAnalysisConfig`; fields and defaults are in Section 4.1.282.
Returns: `MetaAnalysisResult`; field names and types are listed in Section 4.1.283.

3.1.115 tabelz.analysis.missing_value_analysis.compute_missing_value_analysis

```
from tabelz.analysis.missing_value_analysis import compute_missing_value_analysis
compute_missing_value_analysis(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MissingValueConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MissingValueResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MissingValueConfig`; fields and defaults are in Section 4.1.301. **Returns:** `MissingValueResult`; field names and types are listed in Section 4.1.302.

3.1.116 tabelz.analysis.model_based_clustering.compute_model_clustering

```
from tabelz.analysis.model_based_clustering import compute_model_clustering
compute_model_clustering(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ModelClusteringConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ModelClusteringResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ModelClusteringConfig`; fields and defaults are in Section 4.1.321. **Returns:** `ModelClusteringResult`; field names and types are listed in Section 4.1.322.

3.1.117 tabelz.analysis.model_scoring.fit_model_scoring

Fit one supported regression family and retain its scoring artifact.

```
from tabelz.analysis.model_scoring import fit_model_scoring
fit_model_scoring(
    frame: pl.DataFrame | pl.LazyFrame,
    config: ModelScoringConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Any | None = None,
) -> ModelScoringResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ModelScoringConfig`; fields and defaults are in Section 4.1.326. **Returns:** `ModelScoringResult`; field names and types are listed in Section 4.1.329.

3.1.118 tabelz.analysis.model_scoring.score_model_scoring

Score a compatible table using a frozen artifact, without refitting.

```
from tabelz.analysis.model_scoring import score_model_scoring
score_model_scoring(
    artifact: ModelScoringArtifact,
    frame: pl.DataFrame | pl.LazyFrame,
    *,
    row_ids: tuple[str, ...] | None = None,
    excluded_row_ids: tuple[str, ...] = (),
    max_rows: int = 1000000,
    cancel: Any | None = None,
) -> ModelScoringResult
```

artifact
Frozen fitted model returned by fitting or restored from its validated serialized form. Scoring uses this model without refitting.

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

row_ids
Explicit row identities in input order, when supplied. Keep them aligned with the input observations.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `artifact` uses `ModelScoringArtifact`; fields and defaults are in Section 4.1.325. **Returns:** `ModelScoringResult`; field names and types are listed in Section 4.1.329.

3.1.119 tabelz.analysis.model_simulation.compute_model_simulation

Propagate the declared input distributions through the model's conditional mean.

```
from tabelz.analysis.model_simulation import compute_model_simulation
compute_model_simulation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ModelSimulationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ModelSimulationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ModelSimulationConfig`; fields and defaults are in Section 4.1.336. **Returns:** `ModelSimulationResult`; field names and types are listed in Section 4.1.337.

3.1.120 tabelz.analysis.multidimensional_scaling.compute_mds

```
from tabelz.analysis.multidimensional_scaling import compute_mds
compute_mds(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MdsConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MdsResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MdsConfig`; fields and defaults are in Section 4.1.346. **Returns:** `MdsResult`; field names and types are listed in Section 4.1.350.

3.1.121 tabelz.analysis.multiple_correspondence_analysis.compute_mca

```
from tabelz.analysis.multiple_correspondence_analysis import compute_mca
compute_mca(
    frame: pl.LazyFrame | pl.DataFrame,
    config: McaConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> McaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `McaConfig`; fields and defaults are in Section 4.1.356. **Returns:** `McaResult`; field names and types are listed in Section 4.1.358.

3.1.122 tabelz.analysis.multiple_response.compute_mr_crosstab

```
from tabelz.analysis.multiple_response import compute_mr_crosstab
compute_mr_crosstab(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MrCrosstabConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MrCrosstabResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MrCrosstabConfig`; fields and defaults are in Section 4.1.361.
Returns: `MrCrosstabResult`; field names and types are listed in Section 4.1.362.

3.1.123 tabelz.analysis.multiple_response.compute_mr_frequencies

```
from tabelz.analysis.multiple_response import compute_mr_frequencies
compute_mr_frequencies(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MrFrequenciesConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MrFrequenciesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MrFrequenciesConfig`; fields and defaults are in Section 4.1.363.
Returns: `MrFrequenciesResult`; field names and types are listed in Section 4.1.364.

3.1.124 tabelz.analysis.nearest_neighbour.compute_nearest_neighbour

```
from tabelz.analysis.nearest_neighbour import compute_nearest_neighbour
compute_nearest_neighbour(
    frame: pl.LazyFrame | pl.DataFrame,
    config: NearestNeighbourConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> NearestNeighbourResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `NearestNeighbourConfig`; fields and defaults are in Section 4.1.369. **Returns:** `NearestNeighbourResult`; field names and types are listed in Section 4.1.370.

3.1.125 tabelz.analysis.nonlinear_regression.fit_nonlinear

```
from tabelz.analysis.nonlinear_regression import fit_nonlinear
fit_nonlinear(
    frame: pl.LazyFrame,
    config: NonlinearRegressionConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    provenance: dict[str, str | int | float | bool | None] | None = None,
) -> NonlinearRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `NonlinearRegressionConfig`; fields and defaults are in Section 4.1.373. **Returns:** `NonlinearRegressionResult`; field names and types are listed in Section 4.1.374.

3.1.126 tabelz.analysis.nonlinear_regression.predict_nonlinear

```
from tabelz.analysis.nonlinear_regression import predict_nonlinear
predict_nonlinear(
    frame: pl.LazyFrame,
    artifact: NonlinearRegressionArtifact,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> NonlinearRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

artifact
Frozen fitted model returned by fitting or restored from its validated serialized form. Scoring uses this model without refitting.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `artifact` uses `NonlinearRegressionArtifact`; fields and defaults are in Section 4.1.372. **Returns:** `NonlinearRegressionResult`; field names and types are listed in Section 4.1.374.

3.1.127 tabelz.analysis.nonparametric_tolerance_interval

Smallest central order-statistic interval meeting coverage confidence.

```
from tabelz.analysis import nonparametric_tolerance_interval
nonparametric_tolerance_interval(
    values: list[float] | tuple[float, ...] | np.ndarray,
    *,
    content: float = 0.9,
    confidence: float = 0.95,
) -> ToleranceInterval
```

values
One-dimensional observations in their original measurement units; None represents missing input where accepted.

content
Fraction of the population the tolerance interval is intended to contain, on the 0 to 1 scale.

confidence
Confidence probability on the 0 to 1 scale, not a percentage.

Returns: `ToleranceInterval`; field names and types are listed in Section 4.1.280.

3.1.128 tabelz.analysis.normal_capability

Normal capability metrics Cp, Cpk, and mean/stdev for finite data.

```
from tabelz.analysis import normal_capability
normal_capability(
    values: Iterable[float | None],
    lower_spec: float,
    upper_spec: float,
    context: AnalysisContext,
) -> AnalysisOutcome
```

values
One-dimensional observations in their original measurement units; None represents missing input where accepted.

lower_spec
Lower engineering specification limit in the measurement's units.

upper_spec
Upper engineering specification limit in the measurement's units.

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.129 tabelz.analysis.normal_tolerance_interval

Normal-population tolerance interval with explicit content/confidence.

```
from tabelz.analysis import normal_tolerance_interval
normal_tolerance_interval(
    values: list[float] | tuple[float, ...] | np.ndarray,
    *,
    content: float = 0.99,
    confidence: float = 0.95,
    sided: Literal['one_lower', 'one_upper', 'two_sided'] = 'two_sided',
) -> ToleranceInterval
```

values
One-dimensional observations in their original measurement units; None represents missing input where accepted.

content
Fraction of the population the tolerance interval is intended to contain, on the 0 to 1 scale.

confidence
Confidence probability on the 0 to 1 scale, not a percentage.

Returns: `ToleranceInterval`; field names and types are listed in Section 4.1.280.

3.1.130 tabelz.analysis.one_way_anova

Classical one-way fixed-effects ANOVA; requires two non-empty groups.

```
from tabelz.analysis import one_way_anova
one_way_anova(
    groups: Mapping[str, Iterable[float | None]],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.131 tabelz.analysis.operating_characteristic

Return OC-curve points from zero through 100% nonconforming.

```
from tabelz.analysis import operating_characteristic
operating_characteristic(
    sample_size: int,
    accept_number: int,
    *,
    lot_size: int | None = None,
    points: int = 101,
) -> tuple[dict[str, float], ...]
```

3.1.132 tabelz.analysis.ordinary_least_squares

One-predictor OLS with an intercept, using listwise-complete finite rows.

```
from tabelz.analysis import ordinary_least_squares
ordinary_least_squares(
    x: Iterable[float | None],
    y: Iterable[float | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.133 tabelz.analysis.payload_fingerprint

Hash the explicit payload projection, never its storage wrapper.

```
from tabelz.analysis import payload_fingerprint
payload_fingerprint(
    projection: PayloadProjection,
) -> str
```

3.1.134 tabelz.analysis.payload_item_count

Count semantic items: table cells, scatter points, or histogram bins.

```
from tabelz.analysis import payload_item_count
payload_item_count(
    projection: PayloadProjection,
) -> int
```

3.1.135 tabelz.analysis.payload_member_name

Derive the durable content-addressed member path for a payload.

```
from tabelz.analysis import payload_member_name
payload_member_name(
    payload_fingerprint: str,
) -> str
```

3.1.136 tabelz.analysis.payload_ref_from_projection

Build the content-addressed reference for one semantic projection.

```
from tabelz.analysis import payload_ref_from_projection
payload_ref_from_projection(
    projection: PayloadProjection,
) -> PayloadRef
```

Returns: `PayloadRef`; field names and types are listed in Section 4.1.540.

3.1.137 tabelz.analysis.pearson_correlation

Pearson's product-moment correlation on pairwise-complete finite rows.

```
from tabelz.analysis import pearson_correlation
pearson_correlation(
    x: Iterable[float | None],
    y: Iterable[float | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.138 tabelz.analysis.plan_variables_acceptance

Find the smallest one-sided variables plan meeting both risk constraints.

```
from tabelz.analysis import plan_variables_acceptance
plan_variables_acceptance(
    config: VariablesAcceptanceConfig,
    *,
    cancel: Callable[[], bool] | None = None,
) -> VariablesAcceptanceResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `VariablesAcceptanceConfig`; fields and defaults are in Section 4.1.660. **Returns:** `VariablesAcceptanceResult`; field names and types are listed in Section 4.1.661.

3.1.139 tabelz.analysis.plot_tier3_interpolation.compute_contour

```
from tabelz.analysis.plot_tier3_interpolation import compute_contour
compute_contour(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ContourConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ContourResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ContourConfig`; fields and defaults are in Section 4.1.177. **Returns:** `ContourResult`; field names and types are listed in Section 4.1.178.

3.1.140 tabelz.analysis.plot_tier3_interpolation.compute_scatter_3d

```
from tabelz.analysis.plot_tier3_interpolation import compute_scatter_3d
compute_scatter_3d(
    frame: pl.LazyFrame | pl.DataFrame,
    config: Scatter3dConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> Scatter3dResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `Scatter3dConfig`; fields and defaults are in Section 4.1.188.
Returns: `Scatter3dResult`; field names and types are listed in Section 4.1.189.

3.1.141 tabelz.analysis.plot_tier3_preparation.compute_matrix_plot

```
from tabelz.analysis.plot_tier3_preparation import compute_matrix_plot
compute_matrix_plot(
    frame: pl.LazyFrame | pl.DataFrame,
    config: MatrixPlotConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> MatrixPlotResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `MatrixPlotConfig`; fields and defaults are in Section 4.1.180.
Returns: `MatrixPlotResult`; field names and types are listed in Section 4.1.181.

3.1.142 tabelz.analysis.plot_tier3_preparation.compute_stem_and_leaf

```
from tabelz.analysis.plot_tier3_preparation import compute_stem_and_leaf
compute_stem_and_leaf(
    frame: pl.LazyFrame | pl.DataFrame,
    config: StemLeafConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> StemLeafResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `StemLeafConfig`; fields and defaults are in Section 4.1.190. **Returns:** `StemLeafResult`; field names and types are listed in Section 4.1.191.

3.1.143 tabelz.analysis.plot_tier3_probability.compute_probability_plot

```
from tabelz.analysis.plot_tier3_probability import compute_probability_plot
compute_probability_plot(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ProbabilityPlotConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ProbabilityPlotResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ProbabilityPlotConfig`; fields and defaults are in Section 4.1.186. **Returns:** `ProbabilityPlotResult`; field names and types are listed in Section 4.1.187.

3.1.144 tabelz.analysis.predictive_validation.compute_validated_regression

Fit one validated regression with training-only fitted preprocessing.

```
from tabelz.analysis.predictive_validation import compute_validated_regression
compute_validated_regression(
    frame: pl.LazyFrame,
    config: ValidatedRegressionConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ValidatedRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ValidatedRegressionConfig`; fields and defaults are in Section 4.1.401. **Returns:** `ValidatedRegressionResult`; field names and types are listed in Section 4.1.402.

3.1.145 tabelz.analysis.principal_components.compute_principal_components

Compute bounded, deterministic PCA over complete finite cases.

```
from tabelz.analysis.principal_components import compute_principal_components
compute_principal_components(
    frame: pl.LazyFrame,
    config: PcaConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> PcaResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `PcaConfig`; fields and defaults are in Section 4.1.403. **Returns:** `PcaResult`; field names and types are listed in Section 4.1.404.

3.1.146 tabelz.analysis.probability_of_acceptance

Return the OC-curve acceptance probability for a defect fraction.

```
from tabelz.analysis import probability_of_acceptance
probability_of_acceptance(
    defect_fraction: float,
    sample_size: int,
    accept_number: int,
    *,
    lot_size: int | None = None,
) -> float
```

3.1.147 tabelz.analysis.profile_definition_fingerprint

Hash the explicit profile-definition projection.

```
from tabelz.analysis import profile_definition_fingerprint
profile_definition_fingerprint(
    name: str,
    profile_version: str,
    settings: Iterable[ProfileSetting],
) -> str
```

3.1.148 tabelz.analysis.quantile_regression.compute_quantile_regression

Fit every requested quantile and its declared bootstrap uncertainty (999 resamples).

```
from tabelz.analysis.quantile_regression import compute_quantile_regression
compute_quantile_regression(
    frame: pl.LazyFrame | pl.DataFrame,
    config: QuantileRegressionConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> QuantileRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `QuantileRegressionConfig`; fields and defaults are in Section 4.1.430. **Returns:** `QuantileRegressionResult`; field names and types are listed in Section 4.1.431.

3.1.149 tabelz.analysis.rater_agreement.compute_rater_agreement

```
from tabelz.analysis.rater_agreement import compute_rater_agreement
compute_rater_agreement(
    frame: pl.LazyFrame | pl.DataFrame,
    config: RaterAgreementConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> RaterAgreementResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RaterAgreementConfig`; fields and defaults are in Section 4.1.436. **Returns:** `RaterAgreementResult`; field names and types are listed in Section 4.1.437.

3.1.150 tabelz.analysis.reapply_block

Keep block identity while replacing its exact editable configuration.

```
from tabelz.analysis import reapply_block
reapply_block(
    block: ResultTableBlock | PlotBlock | MethodNoteBlock,
    config: SheetAnalysisConfig,
) -> ResultTableBlock | PlotBlock | MethodNoteBlock
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `SheetAnalysisConfig`; fields and defaults are in Section 4.1.553.

3.1.151 tabelz.analysis.registered_lineage

Build registered lineage for an exact base with no divergences.

```
from tabelz.analysis import registered_lineage
registered_lineage(
    base: RegisteredRunProfile | ProfileDefinition,
) -> RegisteredLineage
```

Returns: `RegisteredLineage`; field names and types are listed in Section 4.1.423.

3.1.152 tabelz.analysis.regression_blocks.compute_regression_blocks

```
from tabelz.analysis.regression_blocks import compute_regression_blocks
compute_regression_blocks(
    frame: pl.LazyFrame | pl.DataFrame,
    config: RegressionBlocksConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> RegressionBlocksResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RegressionBlocksConfig`; fields and defaults are in Section 4.1.442. **Returns:** `RegressionBlocksResult`; field names and types are listed in Section 4.1.443.

3.1.153 tabelz.analysis.regression_forest.compute_regression_forest

Run the fixed F106 forest study (500 trees) on the included independent rows.

```
from tabelz.analysis.regression_forest import compute_regression_forest
compute_regression_forest(
    frame: pl.LazyFrame | pl.DataFrame,
    config: RegressionForestConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> RegressionForestResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RegressionForestConfig`; fields and defaults are in Section 4.1.451. **Returns:** `RegressionForestResult`; field names and types are listed in Section 4.1.452.

3.1.154 tabelz.analysis.reliability_scale.compute_scale_reliability

```
from tabelz.analysis.reliability_scale import compute_scale_reliability
compute_scale_reliability(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ScaleReliabilityConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ScaleReliabilityResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ScaleReliabilityConfig`; fields and defaults are in Section 4.1.461. **Returns:** `ScaleReliabilityResult`; field names and types are listed in Section 4.1.462.

3.1.155 tabelz.analysis.render_difference_finder_python

Render deterministic replay code; callers supply their two population mappings.

```
from tabelz.analysis import render_difference_finder_python
render_difference_finder_python(
    *,
    options: DifferenceFinderOptions,
    first_population_id: str,
```

```
    second_population_id: str,
    specifications: Mapping[str, Specification] | None = None,
) -> str
```

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

Argument contract: `options` uses `DifferenceFinderOptions`; fields and defaults are in Section 4.1.112.

3.1.156 tabelz.analysis.render_markdown

Render a deterministic, bounded Markdown view without hidden I/O.

```
from tabelz.analysis import render_markdown
render_markdown(
    result: AnalysisResult,
    receipt: ProvenanceReceipt | None = None,
    resolver: PayloadResolver | None = None,
) -> str
```

result
Typed result returned by the corresponding analysis.
receipt
Provenance receipt paired with the analysis result; None omits receipt rendering when supported.

Argument contract: `result` uses `AnalysisResult`; fields and defaults are in Section 4.1.518.

3.1.157 tabelz.analysis.render_plain_text

Render a deterministic, bounded plain-text view without hidden I/O.

```
from tabelz.analysis import render_plain_text
render_plain_text(
    result: AnalysisResult,
    receipt: ProvenanceReceipt | None = None,
    resolver: PayloadResolver | None = None,
) -> str
```

result
Typed result returned by the corresponding analysis.
receipt
Provenance receipt paired with the analysis result; None omits receipt rendering when supported.

Argument contract: `result` uses `AnalysisResult`; fields and defaults are in Section 4.1.518.

3.1.158 tabelz.analysis.render_selection_explainer_python

Render replay source; callers supply two populations and "context".

```
from tabelz.analysis import render_selection_explainer_python
render_selection_explainer_python(
```

```
    *,
    options: SelectionExplainerOptions,
    selected_population_id: str,
    comparison_population_id: str,
) -> str
```

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

Argument contract: `options` uses `SelectionExplainerOptions`; fields and defaults are in Section 4.1.546.

3.1.159 tabelz.analysis.repeated_covariance.compute_repeated_covariance

Fit one repeated-measures covariance model with Satterthwaite inference.

```
from tabelz.analysis.repeated_covariance import compute_repeated_covariance
compute_repeated_covariance(
    frame: pl.DataFrame | pl.LazyFrame,
    config: RepeatedCovarianceConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> RepeatedCovarianceResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RepeatedCovarianceConfig`; fields and defaults are in Section 4.1.473. **Returns:** `RepeatedCovarianceResult`; field names and types are listed in Section 4.1.474.

3.1.160 tabelz.analysis.repeated_measures.compute_repeated_measures

Implementation 1.1.0 (#111): Helmert contrasts under Automatic for unordered factors, follow-ups at the parent's covariate reference and unavailable follow-up rows kept with their reason. "legacy" replays results recorded at 1.0.0.

```
from tabelz.analysis.repeated_measures import compute_repeated_measures
compute_repeated_measures(
    frame: pl.LazyFrame | pl.DataFrame,
```

```
    config: RepeatedMeasuresConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
    legacy: bool = False,
) -> RepeatedMeasuresResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `RepeatedMeasuresConfig`; fields and defaults are in Section 4.1.489. **Returns:** `RepeatedMeasuresResult`; field names and types are listed in Section 4.1.490.

3.1.161 tabelz.analysis.replace_sheet_block

Replace exactly one existing block without changing its durable identity.

```
from tabelz.analysis import replace_sheet_block
replace_sheet_block(
    sheet: AnalysisSheet,
    replacement: SheetBlock,
) -> AnalysisSheet
```

Argument contract: `sheet` uses `AnalysisSheet`; fields and defaults are in Section 4.1.547. **Returns:** `AnalysisSheet`; field names and types are listed in Section 4.1.547.

3.1.162 tabelz.analysis.replay_forecast_study

Replay a serialized study and optionally require the same input fingerprint.

```
from tabelz.analysis import replay_forecast_study
replay_forecast_study(
    frame: pl.DataFrame,
    specification: dict[str, object],
    *,
    expected_data_fingerprint: str | None = None,
) -> ForecastStudyResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

Returns: `ForecastStudyResult`; field names and types are listed in Section 4.1.650.

3.1.163 tabelz.analysis.report_custom_tables.compute_custom_table

```
from tabelz.analysis.report_custom_tables import compute_custom_table
compute_custom_table(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CustomTableConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CustomTableResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CustomTableConfig`; fields and defaults are in Section 4.1.496.
Returns: `CustomTableResult`; field names and types are listed in Section 4.1.497.

3.1.164 tabelz.analysis.report_explore.compute_explore

```
from tabelz.analysis.report_explore import compute_explore
compute_explore(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ExploreConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> ExploreResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ExploreConfig`; fields and defaults are in Section 4.1.508.
Returns: `ExploreResult`; field names and types are listed in Section 4.1.512.

3.1.165 tabelz.analysis.report_listings.compute_case_summaries

```
from tabelz.analysis.report_listings import compute_case_summaries
compute_case_summaries(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CaseSummariesConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CaseSummariesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CaseSummariesConfig`; fields and defaults are in Section 4.1.501.
Returns: `CaseSummariesResult`; field names and types are listed in Section 4.1.502.

3.1.166 tabelz.analysis.report_listings.compute_codebook

```
from tabelz.analysis.report_listings import compute_codebook
compute_codebook(
    frame: pl.LazyFrame | pl.DataFrame,
    config: CodebookConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> CodebookResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `CodebookConfig`; fields and defaults are in Section 4.1.503.
Returns: `CodebookResult`; field names and types are listed in Section 4.1.505.

3.1.167 tabelz.analysis.report_listings.compute_frequencies

```
from tabelz.analysis.report_listings import compute_frequencies
compute_frequencies(
    frame: pl.LazyFrame | pl.DataFrame,
    config: FrequenciesConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> FrequenciesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `FrequenciesConfig`; fields and defaults are in Section 4.1.513.
Returns: `FrequenciesResult`; field names and types are listed in Section 4.1.514.

3.1.168 tabelz.analysis.represent_payloads

Choose deterministic inline/reference wrappers in canonical order.

```
from tabelz.analysis import represent_payloads
represent_payloads(
    projections: Sequence[PayloadProjection],
) -> tuple[PayloadWrapper, ...]
```

3.1.169 tabelz.analysis.require_recognized_profile

Return the matching registered definition or refuse execution as registered.

```
from tabelz.analysis import require_recognized_profile
require_recognized_profile(
```

```
    snapshot: RegisteredRunProfile,
    registry: ProfileRegistry,
) -> ProfileDefinition
```

Argument contract: `snapshot` uses `RegisteredRunProfile`; fields and defaults are in Section 4.1.424. **Returns:** `ProfileDefinition`; field names and types are listed in Section 4.1.421.

3.1.170 tabelz.analysis.resolve_payload

Resolve and verify one content-addressed payload reference.

```
from tabelz.analysis import resolve_payload
resolve_payload(
    reference: PayloadRef,
    resolver: PayloadResolver | None,
) -> PayloadProjection
```

Argument contract: `reference` uses `PayloadRef`; fields and defaults are in Section 4.1.540.

3.1.171 tabelz.analysis.resolve_specification

Resolve exactly one context-compatible specification and record the decision.

```
from tabelz.analysis import resolve_specification
resolve_specification(
    specs: Sequence[Specification],
    key: str,
    *,
    as_of: date | None = None,
    units: str | None = None,
    applicability: Mapping[str, str] | None = None,
    revision: str | None = None,
    unit_registry: UnitRegistry = DEFAULT_UNIT_REGISTRY,
) -> ResolvedSpecification
```

Returns: `ResolvedSpecification`; field names and types are listed in Section 4.1.569.

3.1.172 tabelz.analysis.resolve_specifications

Compatibility helper returning the resolved specification only.

```
from tabelz.analysis import resolve_specifications
resolve_specifications(
    specs: Sequence[Specification],
    key: str,
    *,
    as_of: date | None = None,
    units: str | None = None,
    applicability: Mapping[str, str] | None = None,
    revision: str | None = None,
    unit_registry: UnitRegistry = DEFAULT_UNIT_REGISTRY,
) -> Specification
```

Returns: `Specification`; field names and types are listed in Section 4.1.570.

3.1.173 tabelz.analysis.run_advanced_time_series

Run an advanced model under a versioned, replayable specification.

```
from tabelz.analysis import run_advanced_time_series
run_advanced_time_series(
    frame: pl.DataFrame,
    spec: AdvancedTimeSeriesSpec,
) -> AdvancedTimeSeriesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `AdvancedTimeSeriesSpec`; fields and defaults are in Section 4.1.621.

3.1.174 tabelz.analysis.run_advanced_time_series_study

Run rolling and untouched-final validation for GARCH or VAR/VECM.

```
from tabelz.analysis import run_advanced_time_series_study
run_advanced_time_series_study(
    frame: pl.DataFrame,
    spec: AdvancedTimeSeriesStudySpec,
) -> dict[str, object]
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `AdvancedTimeSeriesStudySpec`; fields and defaults are in Section 4.1.622.

3.1.175 tabelz.analysis.run_dynamic_regression

Fit the existing dynamic regression, preserving its public result contract.

```
from tabelz.analysis import run_dynamic_regression
run_dynamic_regression(
    frame: pl.DataFrame,
    spec: DynamicRegressionSpec,
) -> DynamicRegressionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `DynamicRegressionSpec`; fields and defaults are in Section 4.1.647. **Returns:** `DynamicRegressionResult`; field names and types are listed in Section 4.1.646.

3.1.176 tabelz.analysis.run_dynamic_regression_study

Validate dynamic regression at rolling, vintage-aware forecast origins.

```
from tabelz.analysis import run_dynamic_regression_study
run_dynamic_regression_study(
    frame: pl.DataFrame,
    spec: DynamicRegressionStudySpec,
) -> DynamicRegressionStudyResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `DynamicRegressionStudySpec`; fields and defaults are in Section 4.1.649. **Returns:** `DynamicRegressionStudyResult`; field names and types are listed in Section 4.1.648.

3.1.177 tabelz.analysis.run_forecast_study

Run rolling validation, an untouched final evaluation, and a future forecast.

```
from tabelz.analysis import run_forecast_study
run_forecast_study(
    frame: pl.DataFrame,
    spec: ForecastStudySpec,
) -> ForecastStudyResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `ForecastStudySpec`; fields and defaults are in Section 4.1.651. **Returns:** `ForecastStudyResult`; field names and types are listed in Section 4.1.650.

3.1.178 tabelz.analysis.run_hypothesis_test

Run one explicit comparative test against the included rows.

```
from tabelz.analysis import run_hypothesis_test
run_hypothesis_test(
    frame: pl.LazyFrame,
    config: HypothesisTestConfig,
) -> HypothesisResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `HypothesisTestConfig`; fields and defaults are in Section 4.1.218. **Returns:** `HypothesisResult`; field names and types are listed in Section 4.1.217.

3.1.179 tabelz.analysis.run_power_analysis

Calculate prospective power or sample size from standardized effect size.

```
from tabelz.analysis import run_power_analysis
run_power_analysis(
    config: PowerAnalysisConfig,
) -> HypothesisResult
```

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

Argument contract: `config` uses `PowerAnalysisConfig`; fields and defaults are in Section 4.1.219. **Returns:** `HypothesisResult`; field names and types are listed in Section 4.1.217.

3.1.180 tabelz.analysis.scatter_data

Return all points or a deterministic identity-based sample of a large source.

```
from tabelz.analysis import scatter_data
scatter_data(
    frame: pl.LazyFrame,
    x: str,
    y: str,
    *,
    max_points: int = 5000,
    seed: int = 0,
) -> PlotData
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

seed
Integer seed for repeatable randomized work.

Returns: `PlotData`; field names and types are listed in Section 4.1.394.

3.1.181 tabelz.analysis.seasonal_adjust

Apply robust STL/MSTL and return trend, seasonal components, and adjusted values.

```
from tabelz.analysis import seasonal_adjust
seasonal_adjust(
    frame: pl.DataFrame,
    response_column: str,
    periods: tuple[int, ...],
    *,
    time_column: str | None = None,
    robust: bool = True,
) -> SeasonalAdjustmentResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

response_column
Name of the response column in the input table.

time_column
Name of the time/order column. Pass None only when the signature permits it; that function determines how row order is then used.

Returns: `SeasonalAdjustmentResult`; field names and types are listed in Section 4.1.545.

3.1.182 tabelz.analysis.selection_explainer_options_from_payload

```
from tabelz.analysis import selection_explainer_options_from_payload
selection_explainer_options_from_payload(
    payload: Mapping[str, object],
) -> SelectionExplainerOptions
```

Returns: `SelectionExplainerOptions`; field names and types are listed in Section 4.1.546.

3.1.183 tabelz.analysis.settings_fingerprint

Hash the explicit settings-identity projection.

```
from tabelz.analysis import settings_fingerprint
settings_fingerprint(
    settings: Iterable[ProfileSetting],
) -> str
```

3.1.184 tabelz.analysis.spc_between_within.compute_between_within

Compute the between/within I-MR-R/S chart set per stage, or chart the data against the approved limits the configuration embeds (#114).

```
from tabelz.analysis.spc_between_within import compute_between_within
compute_between_within(
    frame: pl.LazyFrame | pl.DataFrame,
    config: BetweenWithinConfig,
    *,
    excluded_row_ids: Collection[str] = (),
```

```
    cancel: Callable[[], bool] | None = None,
) -> BetweenWithinResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `BetweenWithinConfig`; fields and defaults are in Section 4.1.556.
Returns: `BetweenWithinResult`; field names and types are listed in Section 4.1.557.

3.1.185 tabelz.analysis.spc_zone.compute_zone_chart

Standardize each stage's subgroup means and accumulate zone scores, or score them against the approved centre and scale the configuration embeds (#114).

```
from tabelz.analysis.spc_zone import compute_zone_chart
compute_zone_chart(
    frame: pl.LazyFrame | pl.DataFrame,
    config: ZoneChartConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> ZoneChartResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `ZoneChartConfig`; fields and defaults are in Section 4.1.565.
Returns: `ZoneChartResult`; field names and types are listed in Section 4.1.566.

3.1.186 tabelz.analysis.spearman_correlation

Spearman correlation: Pearson correlation of average ranks for ties.

```
from tabelz.analysis import spearman_correlation
spearman_correlation(
    x: Iterable[float | None],
    y: Iterable[float | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294. **Returns:** `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.187 tabelz.analysis.stored_model_contour.compute_overlaid_contour

```
from tabelz.analysis.stored_model_contour import compute_overlaid_contour
compute_overlaid_contour(
    frame: pl.LazyFrame | pl.DataFrame | None,
    config: OverlaidContourConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> OverlaidContourResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `OverlaidContourConfig`; fields and defaults are in Section 4.1.572. **Returns:** `OverlaidContourResult`; field names and types are listed in Section 4.1.573.

3.1.188 tabelz.analysis.survey_calibration.compute_survey_calibration

```
from tabelz.analysis.survey_calibration import compute_survey_calibration
compute_survey_calibration(
    frame: polars.dataframe.frame.DataFrame | polars.lazyframe.frame.LazyFrame,
    config: SurveyCalibrationConfig,
    *,
```

```
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyCalibrationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyCalibrationConfig`; fields and defaults are in Section 4.1.578. **Returns:** `SurveyCalibrationResult`; field names and types are listed in Section 4.1.579.

3.1.189 tabelz.analysis.survey_count_models.compute_survey_count

```
from tabelz.analysis.survey_count_models import compute_survey_count
compute_survey_count(
    frame: polars.dataframe.frame.DataFrame | polars.lazyframe.frame.LazyFrame,
    config: SurveyCountConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyCountResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyCountConfig`; fields and defaults are in Section 4.1.590. **Returns:** `SurveyCountResult`; field names and types are listed in Section 4.1.591.

3.1.190 tabelz.analysis.survey_estimates.compute_survey_estimates

Design-based means and totals, and level proportions and counts.

```
from tabelz.analysis.survey_estimates import compute_survey_estimates
compute_survey_estimates(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyEstimatesConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyEstimatesResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyEstimatesConfig`; fields and defaults are in Section 4.1.585. **Returns:** `SurveyEstimatesResult`; field names and types are listed in Section 4.1.586.

3.1.191 tabelz.analysis.survey_estimates.compute_survey_ratios

Ratios of weighted totals with linearized (or replicate) standard errors.

```
from tabelz.analysis.survey_estimates import compute_survey_ratios
compute_survey_ratios(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyRatioConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyRatioResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyRatioConfig`; fields and defaults are in Section 4.1.588.
Returns: `SurveyRatioResult`; field names and types are listed in Section 4.1.589.

3.1.192 tabelz.analysis.survey_models.compute_survey_cox

Design-based Cox regression (Efron ties) with Binder's (1992) variance.

```
from tabelz.analysis.survey_models import compute_survey_cox
compute_survey_cox(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyCoxConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyCoxResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyCoxConfig`; fields and defaults are in Section 4.1.597.
Returns: `SurveyCoxResult`; field names and types are listed in Section 4.1.598.

3.1.193 tabelz.analysis.survey_models.compute_survey_linear

Design-based linear regression: weighted least squares with a design variance.

```
from tabelz.analysis.survey_models import compute_survey_linear
compute_survey_linear(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyLinearConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyLinearResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyLinearConfig`; fields and defaults are in Section 4.1.599. **Returns:** `SurveyLinearResult`; field names and types are listed in Section 4.1.600.

3.1.194 tabelz.analysis.survey_models.compute_survey_logistic

Design-based binary logistic regression (pseudo-ML, linearized sandwich variance).

```
from tabelz.analysis.survey_models import compute_survey_logistic
compute_survey_logistic(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyLogisticConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyLogisticResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyLogisticConfig`; fields and defaults are in Section 4.1.601. **Returns:** `SurveyLogisticResult`; field names and types are listed in Section 4.1.602.

3.1.195 tabelz.analysis.survey_models.compute_survey_ordinal

Design-based proportional-odds regression (pseudo-ML, linearized sandwich variance).

```
from tabelz.analysis.survey_models import compute_survey_ordinal
compute_survey_ordinal(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyOrdinalConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
```

```
    cancel: Callable[[], bool] | None = None,
) -> SurveyOrdinalResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyOrdinalConfig`; fields and defaults are in Section 4.1.604. **Returns:** `SurveyOrdinalResult`; field names and types are listed in Section 4.1.605.

3.1.196 tabelz.analysis.survey_multinomial.compute_survey_multinomial

```
from tabelz.analysis.survey_multinomial import compute_survey_multinomial
compute_survey_multinomial(
    frame: polars.dataframe.frame.DataFrame | polars.lazyframe.frame.LazyFrame,
    config: SurveyMultinomialConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyMultinomialResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyMultinomialConfig`; fields and defaults are in Section 4.1.608. **Returns:** `SurveyMultinomialResult`; field names and types are listed in Section 4.1.609.

3.1.197 tabelz.analysis.survey_replicate_generation.compute_survey_replicates

```
from tabelz.analysis.survey_replicate_generation import compute_survey_replicates
compute_survey_replicates(
    frame: polars.dataframe.frame.DataFrame | polars.lazyframe.frame.LazyFrame,
    config: SurveyReplicateConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyReplicateResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyReplicateConfig`; fields and defaults are in Section 4.1.610. **Returns:** `SurveyReplicateResult`; field names and types are listed in Section 4.1.611.

3.1.198 tabelz.analysis.survey_tables.compute_survey_crosstab

Weighted frequencies of one column, or a crosstab of two with the Rao-Scott F.

```
from tabelz.analysis.survey_tables import compute_survey_crosstab
compute_survey_crosstab(
    frame: pl.DataFrame | pl.LazyFrame,
    config: SurveyCrosstabConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurveyCrosstabResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurveyCrosstabConfig`; fields and defaults are in Section 4.1.582. **Returns:** `SurveyCrosstabResult`; field names and types are listed in Section 4.1.583.

3.1.199 tabelz.analysis.survival_nonparametric.compute_survival

Estimate grouped Kaplan-Meier curves and the two-group log-rank test.

```
from tabelz.analysis.survival_nonparametric import compute_survival
compute_survival(
    frame: pl.LazyFrame,
    config: SurvivalConfig,
    *,
    row_id_column: str = '__tabelz_row_id',
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> SurvivalResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

row_id_column
Column containing stable row identities. The default is the application's reserved identity column, created by the importer.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `SurvivalConfig`; fields and defaults are in Section 4.1.614. **Returns:** `SurvivalResult`; field names and types are listed in Section 4.1.616.

3.1.200 tabelz.analysis.tagged_value

Tag a plain Python value while preserving its strict JSON type.

```
from tabelz.analysis import tagged_value
tagged_value(
    value: bool | int | float | str | None,
) -> TaggedValue
```

3.1.201 tabelz.analysis.time_series_trend.compute_trend

Fit the chosen trend, its accuracy measures and a labelled extrapolation.

```
from tabelz.analysis.time_series_trend import compute_trend
compute_trend(
    frame: pl.LazyFrame | pl.DataFrame,
    config: TrendConfig,
```

```
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> TrendResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `TrendConfig`; fields and defaults are in Section 4.1.641.
Returns: `TrendResult`; field names and types are listed in Section 4.1.643.

3.1.202 tabelz.analysis.two_population_categorical_comparison

Independent categorical comparison using Pearson's chi-square statistic (no p-value).

```
from tabelz.analysis import two_population_categorical_comparison
two_population_categorical_comparison(
    first: Iterable[str | None],
    second: Iterable[str | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.203 tabelz.analysis.two_population_numeric_comparison

Independent-sample numeric comparison: mean difference and pooled Cohen's d.

```
from tabelz.analysis import two_population_numeric_comparison
two_population_numeric_comparison(
    first: Iterable[float | None],
    second: Iterable[float | None],
    context: AnalysisContext,
) -> AnalysisOutcome
```

context
Analysis identity, population, profile and provenance. The complete construction example precedes this catalog; its row count must describe the supplied population.

Argument contract: `context` uses `AnalysisContext`; fields and defaults are in Section 4.1.294.
Returns: `AnalysisOutcome`; field names and types are listed in Section 4.1.295.

3.1.204 tabelz.analysis.two_stage_least_squares.compute_two_stage

```
from tabelz.analysis.two_stage_least_squares import compute_two_stage
compute_two_stage(
    frame: pl.LazyFrame | pl.DataFrame,
    config: TwoStageConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> TwoStageResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `TwoStageConfig`; fields and defaults are in Section 4.1.658.
Returns: `TwoStageResult`; field names and types are listed in Section 4.1.659.

3.1.205 tabelz.analysis.validate_specifications

Reject duplicate/conflicting records and optionally stale-only keys.

```
from tabelz.analysis import validate_specifications
validate_specifications(
    specs: Sequence[Specification],
    *,
    as_of: date | None = None,
    unit_registry: UnitRegistry = DEFAULT_UNIT_REGISTRY,
) -> None
```

3.1.206 tabelz.analysis.variables_acceptance_probability

Return the OC acceptance probability under the stated sigma convention.

```
from tabelz.analysis import variables_acceptance_probability
variables_acceptance_probability(
    defect_fraction: float,
    sample_size: int,
    acceptance_constant: float,
    *,
    sigma_status: Literal['known', 'unknown'],
```

```
) -> float
```

3.1.207 tabelz.analysis.verify_profile_registration

Report whether a registered snapshot is recognized, unknown, or conflicting.

```
from tabelz.analysis import verify_profile_registration
verify_profile_registration(
    snapshot: RegisteredRunProfile,
    registry: ProfileRegistry,
) -> ProfileRegistrationStatus
```

Argument contract: `snapshot` uses `RegisteredRunProfile`; fields and defaults are in Section 4.1.424.

3.1.208 tabelz.analysis.warranty.compute_warranty_preprocess

Convert the triangle and report the counts and recorded assumptions.

```
from tabelz.analysis.warranty import compute_warranty_preprocess
compute_warranty_preprocess(
    frame: pl.LazyFrame | pl.DataFrame,
    config: WarrantyPreprocessConfig,
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> WarrantyPreprocessResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `WarrantyPreprocessConfig`; fields and defaults are in Section 4.1.665. **Returns:** `WarrantyPreprocessResult`; field names and types are listed in Section 4.1.666.

3.1.209 tabelz.analysis.warranty_prediction.compute_warranty_prediction

Fit the family and predict expected failures and costs through the horizon.

```
from tabelz.analysis.warranty_prediction import compute_warranty_prediction
compute_warranty_prediction(
    frame: pl.LazyFrame | pl.DataFrame,
    config: WarrantyPredictionConfig,
```

```
    *,
    excluded_row_ids: Collection[str] = (),
    cancel: Callable[[], bool] | None = None,
) -> WarrantyPredictionResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `WarrantyPredictionConfig`; fields and defaults are in Section 4.1.671. **Returns:** `WarrantyPredictionResult`; field names and types are listed in Section 4.1.672.

3.1.210 tabelz.analysis.weight_estimation.compute_weight_estimation

```
from tabelz.analysis.weight_estimation import compute_weight_estimation
compute_weight_estimation(
    frame: pl.LazyFrame | pl.DataFrame,
    config: WeightEstimationConfig,
    *,
    excluded_row_ids: tuple[str, ...] = (),
    cancel: Callable[[], bool] | None = None,
) -> WeightEstimationResult
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

config
Typed settings object; construct it with the named fields in the type reference below. Required fields have no default.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

cancel
Optional zero-argument callback returning True to request cancellation. None supplies no cancellation callback.

Argument contract: `config` uses `WeightEstimationConfig`; fields and defaults are in Section 4.1.674. **Returns:** `WeightEstimationResult`; field names and types are listed in Section 4.1.675.

3.2 tabelz.data

3.2.1 tabelz.data.SourceImporter.import_csv

```
instance.import_csv(
    path: Path,
    options: CsvImportOptions,
    catalog: TableCatalog,
    *,
    is_cancelled: Callable[[], bool] | None = None,
    table_id: str | None = None,
    source_is_temporary: bool = False,
    logical_name: str | None = None,
) -> ImportResult
```

path
Filesystem path of the input file.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

catalog
TableCatalog receiving imported tables; close it after use to release importer-owned resources.

is_cancelled
Optional zero-argument cancellation callback; return True to stop work.

Argument contract: `options` uses `CsvImportOptions`; fields and defaults are in Section 4.3.1.
Argument contract: `catalog` uses `TableCatalog`; fields and defaults are in Section 4.3.8.
Returns: `ImportResult`; field names and types are listed in Section 4.3.3.

3.2.2 tabelz.data.SourceImporter.import_sql

```
instance.import_sql(
    path: Path,
    options: SqlImportOptions,
    catalog: TableCatalog,
    *,
    is_cancelled: Callable[[], bool] | None = None,
    table_id: str | None = None,
    source_is_temporary: bool = False,
    logical_name: str | None = None,
) -> ImportResult
```

path
Filesystem path of the input file.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

catalog
TableCatalog receiving imported tables; close it after use to release importer-owned resources.

is_cancelled
Optional zero-argument cancellation callback; return True to stop work.

Argument contract: `options` uses `SqlImportOptions`; fields and defaults are in Section 4.3.5.
Argument contract: `catalog` uses `TableCatalog`; fields and defaults are in Section 4.3.8.
Returns: `ImportResult`; field names and types are listed in Section 4.3.3.

3.2.3 tabelz.data.SourceImporter.preview_csv

```
instance.preview_csv(
    path: Path,
    options: CsvImportOptions,
    *,
    is_cancelled: Callable[[], bool] | None = None,
) -> CsvPreview
```

path
Filesystem path of the input file.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

is_cancelled
Optional zero-argument cancellation callback; return True to stop work.

Argument contract: `options` uses `CsvImportOptions`; fields and defaults are in Section 4.3.1.
Returns: `CsvPreview`; field names and types are listed in Section 4.3.2.

3.2.4 tabelz.data.SourceImporter.preview_sql

```
instance.preview_sql(
    path: Path,
    options: SqlImportOptions,
    *,
    is_cancelled: Callable[[], bool] | None = None,
) -> SqlPreview
```

path
Filesystem path of the input file.

options
Import or analysis options of the exact type in the signature; its fields and defaults are listed in the type reference.

is_cancelled
Optional zero-argument cancellation callback; return True to stop work.

Argument contract: `options` uses `SqlImportOptions`; fields and defaults are in Section 4.3.5.
Returns: `SqlPreview`; field names and types are listed in Section 4.3.6.

3.2.5 tabelz.data.TableCatalog.close

Release all importer-owned resources without touching user sources.

```
instance.close(
) -> None
```

3.2.6 tabelz.data.TableCatalog.empty

```
TableCatalog.empty(
) -> TableCatalog
```

Returns: `TableCatalog`; field names and types are listed in Section 4.3.8.

3.3 tabelz.operations

3.3.1 tabelz.operations.FoundationOperationRuntime.__init__

Initialize self. See help(type(self)) for accurate signature.

```
FoundationOperationRuntime(
    registry: OperationRegistry | None = None,
    *,
    model_loader: ModelLoader | None = None,
) -> None
```

3.3.2 tabelz.operations.FoundationOperationRuntime.execute

```
instance.execute(
    frame: pl.LazyFrame,
    spec: OperationSpec,
) -> OperationExecution
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `OperationSpec`; fields and defaults are in Section 4.2.15. **Returns:** `OperationExecution`; field names and types are listed in Section 4.5.3.

3.3.3 tabelz.operations.FoundationOperationRuntime.execute_graph

Execute a typed graph, binding each root operation to its supplied source frame.

```
instance.execute_graph(
    graph: OperationGraph,
    *,
    root_frames: Mapping[str, pl.LazyFrame | Sequence[pl.LazyFrame]],
) -> GraphExecution
```

Argument contract: `graph` uses `OperationGraph`; fields and defaults are in Section 4.5.7. **Returns:** `GraphExecution`; field names and types are listed in Section 4.5.2.

3.3.4 tabelz.operations.FoundationOperationRuntime.execute_many

Execute multi-table joins and concatenations with explicit validation.

```
instance.execute_many(
    frames: Sequence[pl.LazyFrame],
    spec: OperationSpec,
) -> OperationExecution
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `OperationSpec`; fields and defaults are in Section 4.2.15. **Returns:** `OperationExecution`; field names and types are listed in Section 4.5.3.

3.3.5 tabelz.operations.apply_column_formulas

Recalculate all formula targets in dependency order.

```
from tabelz.operations import apply_column_formulas
apply_column_formulas(
    frame: pl.LazyFrame,
    definitions: Mapping[str, ColumnFormula],
    *,
    metadata: TableMetadata | None = None,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.6 tabelz.operations.build_deduplicate

```
from tabelz.operations import build_deduplicate
build_deduplicate(
    columns: list[str],
    *,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.7 tabelz.operations.build_filter

```
from tabelz.operations import build_filter
build_filter(
    predicate: dict[str, Any],
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.8 tabelz.operations.build_group_summary

```
from tabelz.operations import build_group_summary
build_group_summary(
    columns: list[str],
    aggregations: list[dict[str, str]],
    *,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.9 tabelz.operations.build_recode

```
from tabelz.operations import build_recode
build_recode(
    column: str,
    mapping: dict[str, str],
    *,
    unmapped: str = 'keep',
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.10 tabelz.operations.classification_tree_scores

```
from tabelz.operations import classification_tree_scores
classification_tree_scores(
    names: list[str],
    model: dict[str, Any],
    source_table_id: str,
    *,
    target_table_id: str | None = None,
    target_revision: int | None = None,
    target_row_ids: tuple[str, ...] | None = None,
    target_excluded_row_ids: tuple[str, ...] | None = None,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.11 tabelz.operations.compile_formula_expression

Compile a validated formula AST to its declared numeric, text or date output.

```
from tabelz.operations import compile_formula_expression
compile_formula_expression(
    expression: object,
    columns: Sequence[str] | Mapping[str, pl.DataType],
    *,
    mode: FormulaMode = 'numeric',
```

```
    user_missing: Mapping[str, UserMissing] | None = None,
    reference_date: str | date | None = None,
    formula_version: int = 4,
) -> pl.Expr
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

3.3.12 tabelz.operations.computed_column

```
from tabelz.operations import computed_column
computed_column(
    name: str,
    expression: dict[str, Any],
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.13 tabelz.operations.concatenate

```
from tabelz.operations import concatenate
concatenate(
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.14 tabelz.operations.deduplicate

```
from tabelz.operations import deduplicate
deduplicate(
    frame: pl.LazyFrame,
    *,
    subset: Sequence[str],
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.15 tabelz.operations.deterministic_sample

```
from tabelz.operations import deterministic_sample
deterministic_sample(
    frame: pl.LazyFrame,
    *,
    fraction: float,
    seed: int,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

seed
Integer seed for repeatable randomized work.

3.3.16 tabelz.operations.drop

```
from tabelz.operations import drop
drop(
    columns: list[str],
    *,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.17 tabelz.operations.factor_scores

Build a versioned operation carrying its fixed basis and source membership.

```
from tabelz.operations import factor_scores
factor_scores(
    names: list[str],
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.18 tabelz.operations.filter_expression

```
from tabelz.operations import filter_expression
filter_expression(
    frame: pl.LazyFrame,
    predicate: pl.Expr,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.19 tabelz.operations.format_formula

Render a deterministic readable formula, primarily after column rename.

```
from tabelz.operations import format_formula
format_formula(
    expression: object,
```

```
    *,
    mode: FormulaMode = 'numeric',
) -> str
```

3.3.20 tabelz.operations.formula_column

Build a bounded operation that replaces one column with a formula result.

```
from tabelz.operations import formula_column
formula_column(
    target: str,
    expression: dict[str, Any],
    *,
    mode: Literal['numeric', 'text'] = 'numeric',
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.21 tabelz.operations.formula_order

Topologically order formulas and reject direct or indirect cycles.

```
from tabelz.operations import formula_order
formula_order(
    definitions: Mapping[str, ColumnFormula],
) -> tuple[str, ...]
```

3.3.22 tabelz.operations.formula_references

Return referenced column names in first-use order.

```
from tabelz.operations import formula_references
formula_references(
    expression: object,
    *,
    mode: FormulaMode = 'numeric',
) -> tuple[str, ...]
```

3.3.23 tabelz.operations.formula_term_count

Return the bounded operand count used by the UI.

```
from tabelz.operations import formula_term_count
formula_term_count(
    expression: object,
    *,
    mode: FormulaMode = 'numeric',
) -> int
```

3.3.24 tabelz.operations.foundation_registry

Create the registry of typed operations currently executable by Gate A.

```
from tabelz.operations import foundation_registry
foundation_registry(
) -> OperationRegistry
```

Returns: `OperationRegistry`; field names and types are listed in Section 4.5.10.

3.3.25 tabelz.operations.group_summary

```
from tabelz.operations import group_summary
group_summary(
    frame: pl.LazyFrame,
    *,
    by: Sequence[str],
    aggregations: Sequence[pl.Expr],
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.26 tabelz.operations.hierarchical_cluster_labels

Build a versioned operation carrying its fixed cut and source membership.

```
from tabelz.operations import hierarchical_cluster_labels
hierarchical_cluster_labels(
    name: str,
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.27 tabelz.operations.item_analysis_scores

Build a versioned operation carrying its fixed design and source membership.

```
from tabelz.operations import item_analysis_scores
item_analysis_scores(
    names: list[str],
    aggregation: str,
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.28 tabelz.operations.join

```
from tabelz.operations import join
join(
    left_on: str | list[str],
    right_on: str | list[str],
    *,
    how: str = 'inner',
    cardinality: str = 'm:m',
    nulls_equal: bool = False,
    suffix: str = '_right',
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.29 tabelz.operations.keep_per_group

```
from tabelz.operations import keep_per_group
keep_per_group(
    columns: list[str],
    keep: str,
    *,
    order_by: str | None = None,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.30 tabelz.operations.kmeans_cluster_labels

Build a versioned operation carrying its fixed fit and source membership.

```
from tabelz.operations import kmeans_cluster_labels
kmeans_cluster_labels(
    name: str,
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.31 tabelz.operations.model_scoring_scores

Build a versioned score operation carrying immutable artifact JSON.

```
from tabelz.operations import model_scoring_scores
model_scoring_scores(
    names: list[str],
    model: dict[str, Any],
    source_table_id: str,
    *,
```

```
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.32 tabelz.operations.parse_formula

Parse bounded formula text into a validated canonical expression tree.

```
from tabelz.operations import parse_formula
parse_formula(
    text: str,
    *,
    mode: FormulaMode = 'numeric',
    formula_version: int = 4,
) -> FormulaExpression
```

3.3.33 tabelz.operations.principal_component_scores

Build a versioned operation carrying its fixed basis and source membership.

```
from tabelz.operations import principal_component_scores
principal_component_scores(
    names: list[str],
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.34 tabelz.operations.quote_formula_column

Return an exact bracketed column token suitable for insertion.

```
from tabelz.operations import quote_formula_column
quote_formula_column(
    name: str,
) -> str
```

3.3.35 tabelz.operations.recode

```
from tabelz.operations import recode
recode(
    frame: pl.LazyFrame,
    column: str,
    mapping: Mapping[str, str],
    *,
    unmapped: str = 'keep',
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.36 tabelz.operations.regression_tree_scores

Build a versioned operation carrying the fixed artifact and row membership.

```
from tabelz.operations import regression_tree_scores
regression_tree_scores(
    names: list[str],
    model: dict[str, Any],
    source_table_id: str,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.37 tabelz.operations.rename

```
from tabelz.operations import rename
rename(
    mapping: dict[str, str],
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.38 tabelz.operations.rename_columns

```
from tabelz.operations import rename_columns
rename_columns(
    frame: pl.LazyFrame,
    mapping: Mapping[str, str],
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.39 tabelz.operations.rename_formula_reference

Return a formula AST with one column name changed.

```
from tabelz.operations import rename_formula_reference
rename_formula_reference(
    expression: object,
    old: str,
    new: str,
    *,
    mode: FormulaMode = 'numeric',
) -> FormulaExpression
```

3.3.40 tabelz.operations.render_python

Render a graph as deterministic Python that reconstructs its operations.

```
from tabelz.operations import render_python
render_python(
    graph: OperationGraph,
    *,
    allow_private_paths: bool = False,
    migrations: OperationVersionMigrationRegistry | None = None,
) -> str
```

Argument contract: `graph` uses `OperationGraph`; fields and defaults are in Section 4.5.7.

3.3.41 tabelz.operations.reorder

```
from tabelz.operations import reorder
reorder(
    columns: list[str],
    *,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.42 tabelz.operations.reorder_columns

```
from tabelz.operations import reorder_columns
reorder_columns(
    frame: pl.LazyFrame,
    columns: Sequence[str],
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

3.3.43 tabelz.operations.reshape_pivot

Build an immutable root operation for one pivot request.

```
from tabelz.operations import reshape_pivot
reshape_pivot(
    spec: PivotSpec,
    *,
    source_table_id: str,
    output_table_id: str | None = None,
    output_table_name: str = 'Reshaped dataset',
    limits: ReshapeLimits | None = None,
    excluded_row_ids: tuple[str, ...] = (),
    label: str = 'Pivot measurements',
) -> OperationSpec
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

Argument contract: `spec` uses `PivotSpec`; fields and defaults are in Section 4.2.33. **Returns:** `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.44 tabelz.operations.reshape_stack

Build an immutable root operation for one stack request.

```
from tabelz.operations import reshape_stack
reshape_stack(
    spec: StackSpec,
    *,
    source_table_id: str,
    output_table_id: str | None = None,
    output_table_name: str = 'Reshaped dataset',
    limits: ReshapeLimits | None = None,
    excluded_row_ids: tuple[str, ...] = (),
    label: str = 'Stack measurements',
) -> OperationSpec
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

excluded_row_ids
Stable string identities to omit from analysis. The default empty tuple requests no explicit exclusions. These are identities, not displayed row numbers.

Argument contract: `spec` uses `StackSpec`; fields and defaults are in Section 4.2.35. **Returns:** `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.45 tabelz.operations.row_subset

```
from tabelz.operations import row_subset
row_subset(
    offset: int = 0,
    length: int | None = None,
    *,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.46 tabelz.operations.sample

```
from tabelz.operations import sample
sample(
    fraction: float,
```

```
    *,
    seed: int,
    label: str | None = None,
) -> OperationSpec
```

seed
Integer seed for repeatable randomized work.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.47 tabelz.operations.select

```
from tabelz.operations import select
select(
    columns: list[str],
    *,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.48 tabelz.operations.select_columns

```
from tabelz.operations import select_columns
select_columns(
    frame: pl.LazyFrame,
    columns: Sequence[str],
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

3.3.49 tabelz.operations.sort

```
from tabelz.operations import sort
sort(
    columns: list[str],
    *,
    descending: bool = False,
    label: str | None = None,
) -> OperationSpec
```

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.3.50 tabelz.operations.sort_columns

```
from tabelz.operations import sort_columns
sort_columns(
    frame: pl.LazyFrame,
    columns: Sequence[str],
    *,
    descending: bool = False,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

columns
Names of the input columns, in the requested order. Strings must match the table schema exactly.

3.3.51 tabelz.operations.subset_rows

```
from tabelz.operations import subset_rows
subset_rows(
    frame: pl.LazyFrame,
    *,
    offset: int = 0,
    length: int | None = None,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

3.3.52 tabelz.operations.table_edits.build_table_edit_spec

Bind one edit to a stable table ID; graph edges carry state revisions.

```
from tabelz.operations.table_edits import build_table_edit_spec
build_table_edit_spec(
    operation_type: str,
    parameters: Mapping[str, Any],
    *,
    input_table_id: str,
    output_table_id: str | None = None,
    label: str | None = None,
) -> OperationSpec
```

Returns: `OperationSpec`; field names and types are listed in Section 4.2.15.

3.4 tabelz.project

3.4.1 tabelz.project.source_snapshots.write_starting_source_snapshot

Create a typed starting-table snapshot with project-specific errors.

```
from tabelz.project.source_snapshots import write_starting_source_snapshot
write_starting_source_snapshot(
```

```
    frame: pl.LazyFrame,
    snapshot_id: str,
    table_id: str,
    revision: int,
    provenance: SourceProvenance,
    cache_root: Path,
    *,
    is_cancelled: Callable[[], bool] | None = None,
    import_settings: JsonObject | None = None,
) -> SourceSnapshotWrite
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

is_cancelled
Optional zero-argument cancellation callback; return True to stop work.

Argument contract: `provenance` uses `SourceProvenance`; fields and defaults are in Section 4.2.30.
Returns: `SourceSnapshotWrite`; field names and types are listed in Section 4.3.11.

3.5 tabelz.recipes

3.5.1 tabelz.recipes.RecipeContext.__init__

Initialize self. See help(type(self)) for accurate signature.

```
RecipeContext(
    source_root: Path,
) -> None
```

3.5.2 tabelz.recipes.RecipeContext.analyze

Record an analysis with its exact effective settings and seed.

```
instance.analyze(
    spec: OperationSpec,
    *,
    inputs: Sequence[TableHandle] = (),
    parent_operation_ids: Sequence[str] = (),
) -> tuple[ResultHandle, ...]
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `OperationSpec`; fields and defaults are in Section 4.2.15.

3.5.3 tabelz.recipes.RecipeContext.run

Execute the selected closure in the existing shared engine.

```
instance.run(
    *,
    target_set: RecipeTargetSet | None = None,
```

```
    selected_targets: Sequence[str] | None = None,
) -> RecipeReplayPayload
```

Returns: `RecipeReplayPayload`; field names and types are listed in Section 4.4.1.

3.5.4 tabelz.recipes.RecipeContext.run_analysis

Record a typed, allowlisted analysis against one exact table revision.

```
instance.run_analysis(
    request: AnalysisRecipeRequest,
    *,
    table: TableHandle,
) -> ResultHandle
```

Argument contract: `request` uses `AnalysisRecipeRequest`; fields and defaults are in Section 4.5.1. **Argument contract:** `table` uses `TableHandle`; fields and defaults are in Section 4.7.2. **Returns:** `ResultHandle`; field names and types are listed in Section 4.7.1.

3.5.5 tabelz.recipes.RecipeContext.source

Declare one embedded starting table with exact schema and row IDs.

```
instance.source(
    metadata: SourceSnapshotMetadata,
) -> TableHandle
```

Argument contract: `metadata` uses `SourceSnapshotMetadata`; fields and defaults are in Section 4.2.31. **Returns:** `TableHandle`; field names and types are listed in Section 4.7.2.

3.5.6 tabelz.recipes.RecipeContext.transform

Record one table transformation using its registered engine version.

```
instance.transform(
    spec: OperationSpec,
    *,
    inputs: Sequence[TableHandle],
    outputs: Sequence[TableStateRef],
    parent_operation_ids: Sequence[str] = (),
) -> tuple[TableHandle, ...]
```

spec
Versioned specification, carrying the operation or study settings rather than an arbitrary Python callable.

Argument contract: `spec` uses `OperationSpec`; fields and defaults are in Section 4.2.15.

3.5.7 tabelz.recipes.export_recipe

Write a portable folder, publishing it only after every member succeeds.

```
from tabelz.recipes import export_recipe
export_recipe(
    destination: Path,
```

```
    graph: OperationGraph,
    snapshots: Sequence[SourceSnapshotMetadata],
    target_set: RecipeTargetSet,
    members: Mapping[str, MemberPayload],
    *,
    selected_targets: Sequence[str] | None = None,
    member_dependencies: Mapping[str, Sequence[str]] | None = None,
    cancelled: Callable[[], bool] = lambda: False,
) -> Path
```

destination
Filesystem output path. Check the function's overwrite and directory requirements before calling.

cancelled
Zero-argument cancellation callback; return True to stop work.

Argument contract: `graph` uses `OperationGraph`; fields and defaults are in Section 4.5.7.
Argument contract: `target_set` uses `RecipeTargetSet`; fields and defaults are in Section 4.2.27.

3.5.8 tabelz.recipes.write_outputs

Write named table and result outputs under the caller's directory.

```
from tabelz.recipes import write_outputs
write_outputs(
    replay: RecipeReplayPayload,
    destination: Path,
    *,
    target_set: RecipeTargetSet | None = None,
    selected_targets: Sequence[str] | None = None,
) -> tuple[Path, ...]
```

destination
Filesystem output path. Check the function's overwrite and directory requirements before calling.

Argument contract: `replay` uses `RecipeReplayPayload`; fields and defaults are in Section 4.4.1.

3.6 tabelz.selection

3.6.1 tabelz.selection.with_source_row_ids

Attach deterministic IDs derived from immutable namespace and source ordinal.

```
from tabelz.selection import with_source_row_ids
with_source_row_ids(
    frame: pl.LazyFrame,
    table_namespace: str,
) -> pl.LazyFrame
```

frame
Input Polars table. Pass the eager or lazy form accepted by the signature. Column names in the configuration refer to this table.

CHAPTER 4

Settings and returned object fields

Construct settings with named arguments, for example the complete calls in the preceding examples. The Default column distinguishes required arguments from defaults. A factory creates a fresh default when the argument is omitted; do not pass the printed factory label as a value. For returned objects these rows describe accessible attributes; callers normally obtain them from the function instead of constructing results. Bounds shown here come from the typed field declarations; additional cross-field and data-dependent checks can reject an otherwise correctly typed call.

4.1 tabelz.analysis

4.1.1 AdditionalBasicTestsConfig

`tabelz.analysis.additional_basic_tests_contracts.AdditionalBasicTestsConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`method`	`Literal['known_sigma_z','one_variance_chi_square','one_poisson_rate','two_poisson_rate']`	`required`
`input_kind`	`Literal['raw','summary']`	`'raw'`
`response_column`	`str|None`	`None`
`count_column`	`str|None`	`None`
`exposure_column`	`str|None`	`None`
`group_column`	`str|None`	`None`
`group_1`	`str|None`	`None`
`group_2`	`str|None`	`None`
`known_sigma`	`float|None`	`None`
`null_mean`	`float|None`	`None`
`null_variance`	`float|None`	`None`
`null_rate`	`float|None`	`None`
`null_ratio`	`float|None`	`None`
`summary_mean`	`float|None`	`None`
`summary_variance`	`float|None`	`None`
`summary_count`	`int|None`	`None`
`summary_exposure`	`float|None`	`None`
`summary_count_2`	`int|None`	`None`
`summary_exposure_2`	`float|None`	`None`

Field	Type	Default
sample_size	int|None	None
variance_display	Literal['variance','sd']	'variance'
exposure_unit	Literal['units','hours','days']	'units'
alternative	Literal['two-sided','less','greater']	'two-sided'
alpha	float	0.05
confidence_level	float|None	None
max_rows	int	1000000

alpha
Gt(gt=0.0); Lt(lt=0.5)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.2 AdditionalBasicTestsResult

tabelz.analysis.additional_basic_tests_contracts.AdditionalBasicTestsResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method	Literal['known_sigma_z','one_variance_chi_square','one_poisson_rate','two_poisson_rate']	required
input_kind	Literal['raw','summary']	required
alternative	Literal['two-sided','less','greater']	required
alpha	float	required
confidence_level	float	required
estimate	float|None	required
standard_error	float|None	required
statistic	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required
lower_label	str	required
upper_label	str	required
null_value	float|None	required
degrees_of_freedom	float|None	required
sample_size	int	required
count	int|None	required
exposure	float|None	required
count_2	int|None	required
exposure_2	float|None	required
rate	float|None	required

Field	Type	Default
rate_2	float|None	required
ratio	float|None	required
variance_display	Literal['variance','sd']	required
exposure_unit	Literal['units','hours','days']	required
source_count	int	required
observations	int	required
missing_count	int	required
excluded_count	int	required
warnings	tuple[str,...]	()
retained_row_ids	tuple[str,...]	()
method_identity	str	'additional_basic_tests.exact_z_variance_poisson'
implementation_version	str	'1.0.0'
population_fingerprint	str|None	None

4.1.3 AdditionalComparisonsConfig

tabelz.analysis.additional_comparisons.AdditionalComparisonsConfig

Strict one-way comparison request.

Field	Type	Default
response_column	str	required
group_column	str	required
family	Literal['dunnett','games_howell']	'dunnett'
control_level	str|None	None
eligible_groups	tuple[str,...]	()
alpha	float	0.05
seed	int	12345
max_rows	int	1000000
max_groups	int	32

alpha
Gt(gt=0.0); Lt(lt=1.0)

seed
Ge(ge=0); Le(le=2147483647)

max_rows
Ge(ge=1); Le(le=1000000)

max_groups
Ge(ge=2); Le(le=32)

4.1.4 AdditionalComparisonsResult

`tabelz.analysis.additional_comparisons.AdditionalComparisonsResult`

JSON-safe simultaneous comparison result.

Field	Type	Default
response_column	str	required
group_column	str	required
family	Literal['dunnett','games_howell']	required
control_level	str|None	required
eligible_groups	tuple[str,...]	required
alpha	float	required
confidence_level	float	required
seed	int	required
comparisons	tuple[dict[str,object],...]	required
group_summaries	tuple[dict[str,object],...]	required
plot_points	tuple[dict[str,object],...]	required
source_count	int	required
observations	int	required
missing_count	int	required
excluded_count	int	required
ineligible_count	int	required
retained_row_ids	tuple[str,...]	required
assumptions	tuple[str,...]	required
warnings	tuple[str,...]	()
population_fingerprint	str|None	None
method	str	'additional_comparisons'
implementation_version	str	'1.0.0'

4.1.5 ComparisonFamilyRecord

`tabelz.analysis.adjusted_mean_families.ComparisonFamilyRecord`

What one factor's max-t family compared and how it was calibrated.

Field	Type	Default
factor	str	required
procedure	Literal['tukey','dunnett']	required
control_level	str|None	required
levels	tuple[str,...]	required
averaged_over	tuple[str,...]	required

Field	Type	Default
covariate_values	dict[str,float]	required
residual_df	int	required
critical_value	float|None	required
critical_bracket	tuple[float,float]|None	required
calibration	FamilyCalibration|None	required
notes	tuple[str,...]	()
alpha	float	0.05

alpha
Gt(gt=0); Lt(lt=0.5)

4.1.6 FamilyCalibration

tabelz.analysis.adjusted_mean_families.FamilyCalibration

The max-t calibration record (#105) in a list-tolerant form.

Field	Type	Default
method	str	required
replications	int	required
points_per_replication	int	required
total_points	int	required
seeds	tuple[int,...]	required
target_half_width	float	required
p_half_width	float	required
critical_probability_half_width	float	required
status	str	required

4.1.7 AnalysisOfMeansConfig

tabelz.analysis.analysis_of_means.AnalysisOfMeansConfig

Which ANOM to run, its columns and alpha (0.05 by default; 0.10 and 0.01 selectable).

Field	Type	Default
analysis	Literal['normal_one_way','normal_two_way','binomial','poisson']	'normal_one_way'
response	str|None	None
group	str|None	None
factor_a	str|None	None
factor_b	str|None	None
events	str|None	None

Field	Type	Default
trials	str|None	None
exposure	str|None	None
exposure_unit	str	'unit'
alpha	Literal['0.10','0.05','0.01']	'0.05'

exposure_unit
MinLen(min_length=1); MaxLen(max_length=40)

4.1.8 AnalysisOfMeansResult

tabelz.analysis.analysis_of_means.AnalysisOfMeansResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
analysis	Literal['normal_one_way','normal_two_way','binomial','poisson']	required
alpha	float	required
response	str|None	required
group	str|None	required
factor_a	str|None	required
factor_b	str|None	required
events	str|None	required
trials	str|None	required
exposure	str|None	required
exposure_unit	str	required
panels	tuple[AnomPanel,...]	required
residual_df	int|None	required
pooled_sd	float|None	required
observations	int	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
warnings	tuple[str,...]	()

observations
Ge(ge=0)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)

4.1.9 AnomGroup

`tabelz.analysis.analysis_of_means.AnomGroup`

One plotted group (or interaction cell) with its limits and decision.

Field	Type	Default
`label`	`str`	`required`
`size`	`float`	`required`
`events`	`float|None`	`None`
`statistic`	`float`	`required`
`standard_error`	`float`	`required`
`lower`	`float`	`required`
`upper`	`float`	`required`
`decision`	`Literal['above','below','within']`	`required`
`rows`	`int`	`required`
`near_limit`	`bool`	`False`

size
Gt(gt=0.0)

standard_error
Ge(ge=0.0)

rows
Ge(ge=0)

4.1.10 AnomPanel

`tabelz.analysis.analysis_of_means.AnomPanel`

One separately calibrated decision chart.

Field	Type	Default
`effect`	`str`	`required`
`kind`	`Literal['means','interaction','proportions','rates']`	`required`
`alpha`	`float`	`required`
`center`	`float`	`required`
`critical_value`	`float`	`required`
`critical_df`	`int|None`	`required`
`critical_method`	`str`	`required`
`critical_standard_error`	`float|None`	`None`
`groups`	`tuple[AnomGroup,...]`	`required`
`signals`	`int`	`required`

critical_value
Gt(gt=0.0)

signals
Ge(ge=0)

4.1.11 AnovaPostHocConfig

`tabelz.analysis.anova_post_hoc.AnovaPostHocConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
group	str	required
procedure	Literal['none','lsd','bonferroni','sidak','holm','scheffe','tukey','gt2','gabriel','dunnett','snk','duncan','regwq','regwf','games_howell','t2','t3','dunnett_c']	'tukey'
alpha	float	0.05
control	str	''
alternative	Literal['two-sided','greater','less']	'two-sided'
contrast_family	Literal['none','custom','deviation','simple_first','simple_last','difference','helmert','repeated','polynomial']	'none'
coefficients	tuple[float,...]	()
level_values	tuple[float,...]	()
contrast_variance	Literal['pooled','welch']	'pooled'
contrast_adjustment	Literal['none','bonferroni','holm','sidak','scheffe']	'none'

alpha
Gt(gt=0.0); Lt(lt=0.5)

4.1.12 AnovaPostHocResult

`tabelz.analysis.anova_post_hoc.AnovaPostHocResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	AnovaPostHocConfig	required
groups	tuple[GroupSummary,...]	required
anova	AnovaTable	required
comparisons	ComparisonResult|None	None
contrasts	ContrastResult|None	None
excluded_count	int	required

Field	Type	Default
missing_count	int	required

4.1.13 AnovaTable

tabelz.analysis.anova_post_hoc.AnovaTable

Common strict, immutable model baseline for public contracts.

Field	Type	Default
between_ss	float	required
within_ss	float	required
between_df	int	required
within_df	int	required
f_value	float	required
p_value	float	required
mse	float	required

4.1.14 AssociationConfig

tabelz.analysis.association_measures.AssociationConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row	str	required
column	str	required
confidence	float	0.95
haldane_correction	bool	False
row_order	tuple[str, ...]	()
column_order	tuple[str, ...]	()

confidence
Gt(gt=0.5); Lt(lt=1.0)

4.1.15 AssociationResult

tabelz.analysis.association_measures.AssociationResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	AssociationConfig	required
rows_analysed	int	required
excluded_count	int	required

Field	Type	Default
missing_count	int	required
row_categories	tuple[str,...]	required
column_categories	tuple[str,...]	required
counts	tuple[tuple[int,...],...]	required
chi_square	float	required
chi_square_df	int	required
chi_square_p	float|None	required
cells_expected_below_5	int	required
measures	tuple[Measure,...]	required
risks	tuple[RiskEstimate,...]	()
cells	tuple[CellDetail,...]	required
warnings	tuple[str,...]	()
row_order_source	Literal['numeric','declared','undeclared']	'numeric'
column_order_source	Literal['numeric','declared','undeclared']	'numeric'

4.1.16 CellDetail

tabelz.analysis.association_measures.CellDetail

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row	str	required
column	str	required
count	int	required
expected	float	required
standardized_residual	float|None	required
adjusted_residual	float|None	required

4.1.17 Measure

tabelz.analysis.association_measures.Measure

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['nominal','ordinal','chi-square','interval']	required
name	str	required

Field	Type	Default
direction	str	required
estimate	float|None	required
ase1	float|None	None
ase0	float|None	None
t_value	float|None	None
p_value	float|None	None
lower	float|None	None
upper	float|None	None
note	str	''

4.1.18 RiskEstimate

`tabelz.analysis.association_measures.RiskEstimate`

A 2 x 2 risk estimate; an infinite boundary estimate sets “infinite“ (no number).

Field	Type	Default
name	str	required
estimate	float|None	required
lower	float|None	required
upper	float|None	required
infinite	bool	False
note	str	''

4.1.19 AgreementEstimate

`tabelz.analysis.attribute_agreement.AgreementEstimate`

Field	Type	Default
agreement_pct	float	required
ci_lower	float	required
ci_upper	float	required
kappa	float|None	required
method	str	required
comparisons	int	required

4.1.20 AppraiserAgreement

`tabelz.analysis.attribute_agreement.AppraiserAgreement`

Field	Type	Default
appraiser	str	required

Field	Type	Default
within	AgreementEstimate	required
versus_standard	AgreementEstimate|None	required

4.1.21 AttributeAgreementConfig

tabelz.analysis.attribute_agreement.AttributeAgreementConfig

Field	Type	Default
rating	str	required
part	str	required
appraiser	str	required
trial	str|None	None
standard	str|None	None
scale	Literal['nominal','ordinal_linear','ordinal_quadratic']	'nominal'

4.1.22 AttributeAgreementResult

tabelz.analysis.attribute_agreement.AttributeAgreementResult

Field	Type	Default
config	AttributeAgreementConfig	required
n_parts	int	required
n_appraisers	int	required
n_trials	int	required
categories	tuple[str,...]	required
complete_rows	int	required
dropped_rows	int	required
between	AgreementEstimate	required
overall_standard	AgreementEstimate|None	required
appraisers	tuple[AppraiserAgreement,...]	required
confusion_columns	tuple[str,...]	required
confusion_rows	tuple[tuple[object,...],...]	required
warnings	tuple[str,...]	required

4.1.23 BayesAnovaConfig

tabelz.analysis.bayesian.contracts.BayesAnovaConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
factor	str	required
prior_scale	float	0.5
confidence_level	float	0.95

prior_scale
Gt(gt=0.0); Le(le=10.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.24 BayesAnovaResult

tabelz.analysis.bayesian.contracts.BayesAnovaResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
procedure	str	required
n	int	required
missing_count	int	required
excluded_count	int	required
priors	tuple[BayesPrior, ...]	required
bayes_factors	tuple[BayesFactor, ...]	required
posteriors	tuple[BayesPosterior, ...]	required
prior_curve	tuple[tuple[float, float], ...]	()
posterior_curve	tuple[tuple[float, float], ...]	()
curve_parameter	str	''
notes	tuple[str, ...]	()
config	BayesAnovaConfig	required

4.1.25 BayesBinomialConfig

tabelz.analysis.bayesian.contracts.BayesBinomialConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
event_value	str	required
null_proportion	float	0.5
prior_a	float	1.0
prior_b	float	1.0
confidence_level	float	0.95

null_proportion
Gt(gt=0.0); Lt(lt=1.0)

prior_a
Gt(gt=0.0)

prior_b
Gt(gt=0.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.26 BayesBinomialResult

`tabelz.analysis.bayesian.contracts.BayesBinomialResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`procedure`	`str`	`required`
`n`	`int`	`required`
`missing_count`	`int`	`required`
`excluded_count`	`int`	`required`
`priors`	`tuple[BayesPrior,...]`	`required`
`bayes_factors`	`tuple[BayesFactor,...]`	`required`
`posteriors`	`tuple[BayesPosterior,...]`	`required`
`prior_curve`	`tuple[tuple[float,float],...]`	`()`
`posterior_curve`	`tuple[tuple[float,float],...]`	`()`
`curve_parameter`	`str`	`''`
`notes`	`tuple[str,...]`	`()`
`config`	`BayesBinomialConfig`	`required`

4.1.27 BayesCorrelationConfig

`tabelz.analysis.bayesian.contracts.BayesCorrelationConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`first`	`str`	`required`
`second`	`str`	`required`
`prior_width`	`float`	`1.0`
`confidence_level`	`float`	`0.95`

prior_width
Gt(gt=0.0); Le(le=10.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.28 BayesCorrelationResult

`tabelz.analysis.bayesian.contracts.BayesCorrelationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`procedure`	`str`	`required`
`n`	`int`	`required`
`missing_count`	`int`	`required`
`excluded_count`	`int`	`required`
`priors`	`tuple[BayesPrior,...]`	`required`
`bayes_factors`	`tuple[BayesFactor,...]`	`required`
`posteriors`	`tuple[BayesPosterior,...]`	`required`
`prior_curve`	`tuple[tuple[float,float],...]`	`()`
`posterior_curve`	`tuple[tuple[float,float],...]`	`()`
`curve_parameter`	`str`	`''`
`notes`	`tuple[str,...]`	`()`
`config`	`BayesCorrelationConfig`	`required`

4.1.29 BayesFactor

`tabelz.analysis.bayesian.contracts.BayesFactor`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`numerator`	`str`	`required`
`denominator`	`str`	`required`
`bf10`	`float|None`	`required`
`bf01`	`float|None`	`required`
`log_bf10`	`float|None`	`required`
`error`	`float|None`	`required`
`posterior_probability`	`float|None`	`None`
`prior_probability`	`float|None`	`None`
`note`	`str`	`''`

4.1.30 BayesNormalConfig

`tabelz.analysis.bayesian.contracts.BayesNormalConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
known_sd	float	required
prior_mean	float	0.0
prior_sd	float	1.0
null_value	float	0.0
confidence_level	float	0.95

known_sd
Gt(gt=0.0)

prior_sd
Gt(gt=0.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.31 BayesNormalResult

tabelz.analysis.bayesian.contracts.BayesNormalResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
procedure	str	required
n	int	required
missing_count	int	required
excluded_count	int	required
priors	tuple[BayesPrior, ...]	required
bayes_factors	tuple[BayesFactor, ...]	required
posteriors	tuple[BayesPosterior, ...]	required
prior_curve	tuple[tuple[float, float], ...]	()
posterior_curve	tuple[tuple[float, float], ...]	()
curve_parameter	str	''
notes	tuple[str, ...]	()
config	BayesNormalConfig	required

4.1.32 BayesPoissonConfig

tabelz.analysis.bayesian.contracts.BayesPoissonConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
exposure	str|None	None

Field	Type	Default
null_rate	float	1.0
prior	Literal['gamma','jeffreys']	'gamma'
prior_shape	float	1.0
prior_rate	float	1.0
confidence_level	float	0.95

null_rate
Gt(gt=0.0)

prior_shape
Gt(gt=0.0)

prior_rate
Gt(gt=0.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.33 BayesPoissonResult

tabelz.analysis.bayesian.contracts.BayesPoissonResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
procedure	str	required
n	int	required
missing_count	int	required
excluded_count	int	required
priors	tuple[BayesPrior,...]	required
bayes_factors	tuple[BayesFactor,...]	required
posteriors	tuple[BayesPosterior,...]	required
prior_curve	tuple[tuple[float,float],...]	()
posterior_curve	tuple[tuple[float,float],...]	()
curve_parameter	str	''
notes	tuple[str,...]	()
config	BayesPoissonConfig	required

4.1.34 BayesPosterior

tabelz.analysis.bayesian.contracts.BayesPosterior

Common strict, immutable model baseline for public contracts.

Field	Type	Default
parameter	str	required
mean	float|None	required

Field	Type	Default
sd	float|None	required
median	float	required
lower	float	required
upper	float	required
hdi_lower	float|None	required
hdi_upper	float|None	required
hdi_note	str	''
moment_note	str	''
numerical_error	float|None	None
moment_error	float|None	None

4.1.35 BayesPrior

tabelz.analysis.bayesian.contracts.BayesPrior

Common strict, immutable model baseline for public contracts.

Field	Type	Default
parameter	str	required
family	str	required
settings	tuple[tuple[str,float],...]	required
note	str	''

4.1.36 BayesRegressionConfig

tabelz.analysis.bayesian.contracts.BayesRegressionConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
predictors	tuple[str,...]	required
models	tuple[tuple[str,...],...]	()
model_prior	tuple[float,...]	()
null_model_prior	float|None	None
prior	Literal['zellner_siow','hyper_g']	'zellner_siow'
prior_scale	float	0.3535533905932738
hyper_a	float	3.0
confidence_level	float	0.95

predictors
MinLen(min_length=1); MaxLen(max_length=10)

prior_scale
Gt(gt=0.0); Le(le=10.0)

hyper_a
Gt(gt=2.0); Le(le=4.0)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.37 BayesRegressionResult

`tabelz.analysis.bayesian.contracts.BayesRegressionResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`procedure`	`str`	`required`
`n`	`int`	`required`
`missing_count`	`int`	`required`
`excluded_count`	`int`	`required`
`priors`	`tuple[BayesPrior,...]`	`required`
`bayes_factors`	`tuple[BayesFactor,...]`	`required`
`posteriors`	`tuple[BayesPosterior,...]`	`required`
`prior_curve`	`tuple[tuple[float,float],...]`	`()`
`posterior_curve`	`tuple[tuple[float,float],...]`	`()`
`curve_parameter`	`str`	`''`
`notes`	`tuple[str,...]`	`()`
`config`	`BayesRegressionConfig`	`required`

4.1.38 BayesTTestConfig

`tabelz.analysis.bayesian.contracts.BayesTTestConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`design`	`Literal['one_sample','paired','independent']`	`'one_sample'`
`column`	`str`	`required`
`second`	`str|None`	`None`
`group`	`str|None`	`None`
`first_group`	`str|None`	`None`
`null_value`	`float`	`0.0`
`prior_scale`	`float`	`0.7071067811865475`
`confidence_level`	`float`	`0.95`

prior_scale
Gt(gt=0.0); Le(le=10.0)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.39 BayesTTestResult

`tabelz.analysis.bayesian.contracts.BayesTTestResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
procedure	str	required
n	int	required
missing_count	int	required
excluded_count	int	required
priors	tuple[BayesPrior,...]	required
bayes_factors	tuple[BayesFactor,...]	required
posteriors	tuple[BayesPosterior,...]	required
prior_curve	tuple[tuple[float,float],...]	()
posterior_curve	tuple[tuple[float,float],...]	()
curve_parameter	str	''
notes	tuple[str,...]	()
config	BayesTTestConfig	required

4.1.40 BlockedRankTestsConfig

`tabelz.analysis.blocked_rank_tests.BlockedRankTestsConfig`

Strict request for a complete-block Friedman or ordered runs test.

Field	Type	Default
method	Literal['friedman','runs']	'friedman'
response_column	str|None	None
treatment_column	str|None	None
block_column	str|None	None
category_column	str|None	None
order_column	str|None	None
reference_mode	Literal['median','fixed']	'median'
reference_value	float|None	None
inference	Literal['asymptotic','exact']	'asymptotic'
alternative	Literal['two-sided','less','greater']	'two-sided'
max_rows	int	1000000
max_blocks	int	10000
max_exact_n	int	200

max_rows
Ge(ge=1); Le(le=1000000)
max_blocks
Ge(ge=1); Le(le=10000)

max_exact_n
Ge(ge=3); Le(le=200)

4.1.41 BlockedRankTestsResult

`tabelz.analysis.blocked_rank_tests.BlockedRankTestsResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
test_kind	Literal['friedman','runs']	required
response_column	str|None	required
treatment_column	str|None	required
block_column	str|None	required
category_column	str|None	required
order_column	str|None	required
inference	Literal['asymptotic','exact']	required
alternative	Literal['two-sided','less','greater']	required
source_count	int	required
observations	int	required
missing_count	int	required
excluded_count	int	required
retained_row_ids	tuple[str,...]	()
population_fingerprint	str	required
friedman_statistic	float|None	None
friedman_degrees_of_freedom	int|None	None
tie_correction	float|None	None
block_count	int|None	None
treatment_count	int|None	None
tie_block_count	int	0
rank_summaries	tuple[dict[str,object],...]	()
reference_value	float|None	None
category_levels	tuple[str,...]	()
sequence_length	int|None	None
category_counts	dict[str,int]	factory:dict
runs	int|None	None
ties_omitted	int	0
expected_runs	float|None	None
runs_variance	float|None	None
runs_z	float|None	None
p_value	float|None	None
warnings	tuple[str,...]	()

Field	Type	Default
method	str	'blocked_rank_tests.friedman_runs'
implementation_version	str	'1.0.0'

4.1.42 BootstrapBin

tabelz.analysis.bootstrap_framework.BootstrapBin

Common strict, immutable model baseline for public contracts.

Field	Type	Default
lower	float	required
upper	float	required
count	int	required

4.1.43 BootstrapComponent

tabelz.analysis.bootstrap_framework.BootstrapComponent

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
attempted	int	required
successful	int	required
failed	int	required
failures	tuple[tuple[str,int],...]	()
withheld	str	''
bootstrap_mean	float|None	None
bias	float|None	None
standard_error	float|None	None
mc_se_bias	float|None	None
mc_se_standard_error	float|None	None
percentile_lower	float|None	None
percentile_upper	float|None	None
percentile_note	str	''
bca_lower	float|None	None
bca_upper	float|None	None
bca_note	str	''
z0	float|None	None

Field	Type	Default
acceleration	float|None	None
bca_probabilities	tuple[float,float]|None	None

4.1.44 BootstrapEstimateConfig

tabelz.analysis.bootstrap_framework.BootstrapEstimateConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
statistic	Literal['mean','median','trimmed_mean','sd','quantile','pearson','spearman','kendall','linear_coefficients','logistic_coefficients','mean_difference','median_difference','cronbach_alpha','kappa','krippendorff_alpha']	'mean'
column	str|None	None
second	str|None	None
columns	tuple[str,...]	()
response	str|None	None
predictors	tuple[str,...]	()
factors	tuple[str,...]	()
event	str|None	None
group	str|None	None
first_group	str|None	None
categories	tuple[str,...]	()
trim	float	0.05
probability	float	0.25
metric	Literal['nominal','ordinal','interval']	'nominal'
scheme	Literal['rows','stratified','clusters','clusters_in_strata']	'rows'
strata	str|None	None
clusters	str|None	None
resamples	int	5000
seed	int	20260928
confidence_level	float	0.95
intervals	Literal['percentile','percentile_and_bca']	'percentile'

trim
Ge(ge=0.0); Lt(lt=0.5)

probability
Gt(gt=0.0); Lt(lt=1.0)

resamples
Ge(ge=2); Le(le=10000)
seed
Ge(ge=0); Le(le=9223372036854775807)
confidence_level
Gt(gt=0.0); Lt(lt=1.0)

4.1.45 BootstrapEstimateResult

`tabelz.analysis.bootstrap_framework.BootstrapEstimateResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`BootstrapEstimateConfig`	`required`
`statistic_label`	`str`	`required`
`assumption`	`str`	`required`
`scheme_label`	`str`	`required`
`rows_used`	`int`	`required`
`missing_count`	`int`	`required`
`excluded_count`	`int`	`required`
`units`	`int`	`required`
`strata_count`	`int`	`required`
`components`	`tuple[BootstrapComponent, ...]`	`required`
`histogram`	`tuple[BootstrapBin, ...]`	`()`
`notes`	`tuple[str, ...]`	`()`
`warnings`	`tuple[str, ...]`	`()`

4.1.46 CanonicalConfig

`tabelz.analysis.canonical_correlation.CanonicalConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`first`	`tuple[str, ...]`	`required`
`second`	`tuple[str, ...]`	`required`

first
MinLen(min_length=1); MaxLen(max_length=50)
second
MinLen(min_length=1); MaxLen(max_length=50)

4.1.47 CanonicalLoading

`tabelz.analysis.canonical_correlation.CanonicalLoading`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
set	str	required
variable	str	required
raw	tuple[float, ...]	required
standardized	tuple[float, ...]	required
own_structure	tuple[float, ...]	required
cross_structure	tuple[float, ...]	required

4.1.48 CanonicalRedundancy

tabelz.analysis.canonical_correlation.CanonicalRedundancy

Common strict, immutable model baseline for public contracts.

Field	Type	Default
index	int	required
first_own_variance	float	required
first_redundancy	float	required
second_own_variance	float	required
second_redundancy	float	required

4.1.49 CanonicalResult

tabelz.analysis.canonical_correlation.CanonicalResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CanonicalConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
roots	tuple[CanonicalRoot, ...]	required
loadings	tuple[CanonicalLoading, ...]	required
redundancy	tuple[CanonicalRedundancy, ...]	required
bartlett_multiplier	float	required
notes	tuple[str, ...]	()

4.1.50 CanonicalRoot

tabelz.analysis.canonical_correlation.CanonicalRoot

Common strict, immutable model baseline for public contracts.

Field	Type	Default
index	int	required
correlation	float	required
squared	float	required
wilks_lambda	float	required
chi_square	float|None	required
df	int	required
p_value	float|None	required

4.1.51 CapabilitySetup

tabelz.analysis.capability_sixpack.CapabilitySetup

One characteristic's specification, capability model and subgrouping.

Field	Type	Default
column	str	required
lsl	float|None	None
usl	float|None	None
target	float|None	None
method	Literal['normal','lognormal','weibull','box_cox']	'normal'
subgroup_mode	Literal['individuals','constant','by_column']	'individuals'
subgroup_size	int	5
group_column	str|None	None
confidence_level	float	0.95

subgroup_size
Ge(ge=2); Le(le=50)
confidence_level
Ge(ge=0.5); Lt(lt=1.0)

4.1.52 ChartPoint

tabelz.analysis.capability_sixpack.ChartPoint

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	float	required
y	float	required
center	float|None	None
upper	float|None	None

Field	Type	Default
lower	float|None	None

4.1.53 ComparisonConfig

tabelz.analysis.capability_sixpack.ComparisonConfig

One characteristic's specification, capability model and subgrouping.

Field	Type	Default
column	str	required
lsl	float|None	None
usl	float|None	None
target	float|None	None
method	Literal['normal','lognormal','weibull','box_cox']	'normal'
subgroup_mode	Literal['individuals','constant','by_column']	'individuals'
subgroup_size	int	5
group_column	str|None	None
confidence_level	float	0.95
split_kind	Literal['group','point']	'group'
split_column	str	required
split_value	float|None	None

subgroup_size
Ge(ge=2); Le(le=50)

confidence_level
Ge(ge=0.5); Lt(lt=1.0)

4.1.54 ComparisonResult

tabelz.analysis.capability_sixpack.ComparisonResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ComparisonConfig	required
populations	tuple[PopulationRow,...]	required

4.1.55 MultipleConfig

tabelz.analysis.capability_sixpack.MultipleConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
columns	tuple[str,...]	required
lsls	tuple[str,...]	()
usls	tuple[str,...]	()
targets	tuple[str,...]	()
method	Literal['normal','lognormal','weibull','box_cox']	'normal'
subgroup_mode	Literal['individuals','constant','by_column']	'individuals'
subgroup_size	int	5
group_column	str|None	None
confidence_level	float	0.95

columns
MinLen(min_length=1)

subgroup_size
Ge(ge=2); Le(le=50)

confidence_level
Ge(ge=0.5); Lt(lt=1.0)

4.1.56 MultipleResult

tabelz.analysis.capability_sixpack.MultipleResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MultipleConfig	required
rows	tuple[JsonObject,...]	required

4.1.57 PopulationRow

tabelz.analysis.capability_sixpack.PopulationRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
population	str	required
rows	int	required
excluded	int	required
capability	JsonObject	required

4.1.58 SixpackResult

tabelz.analysis.capability_sixpack.SixpackResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CapabilitySetup	required
capability	JsonObject	required
chart_family	str	required
primary_title	str	required
primary	tuple[ChartPoint, ...]	required
secondary_title	str	required
secondary	tuple[ChartPoint, ...]	required
run_plot	tuple[ChartPoint, ...]	required
run_plot_subgroups	int	required
histogram	tuple[tuple[float, float, int], ...]	required
overall_curve	tuple[tuple[float, float], ...]	required
within_curve	tuple[tuple[float, float], ...]	required
within_spread	tuple[float, float]|None	required
overall_spread	tuple[float|None, float|None]|None	required
excluded_count	int	required
overall_spread_label	str	'mean+/-3overallSD'
curve_note	str	''

4.1.59 CatpcaArtifact

tabelz.analysis.categorical_pca.CatpcaArtifact

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variables	tuple[CatpcaQuantified, ...]	required
vectors	tuple[tuple[float, ...], ...]	required
eigenvalues	tuple[float, ...]	required

4.1.60 CatpcaCategory

tabelz.analysis.categorical_pca.CatpcaCategory

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variable	str	required
category	str	required
count	int	required
quantification	float	required

4.1.61 CatpcaConfig

`tabelz.analysis.categorical_pca.CatpcaConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`nominal`	`tuple[str,...]`	`()`
`ordinal`	`tuple[str,...]`	`()`
`numeric`	`tuple[str,...]`	`()`
`orders`	`tuple[str,...]`	`()`
`dimensions`	`int`	`2`
`max_iterations`	`int`	`1000`

dimensions
Ge(ge=1); Le(le=10)
max_iterations
Ge(ge=1); Le(le=5000)

4.1.62 CatpcaDimension

`tabelz.analysis.categorical_pca.CatpcaDimension`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`dimension`	`int`	`required`
`eigenvalue`	`float`	`required`
`percent`	`float`	`required`
`alpha`	`float`	`required`

4.1.63 CatpcaQuantified

`tabelz.analysis.categorical_pca.CatpcaQuantified`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`variable`	`str`	`required`
`level`	`str`	`required`
`categories`	`tuple[str,...]`	`()`
`quantifications`	`tuple[float,...]`	`()`
`mean`	`float|None`	`None`
`sd`	`float|None`	`None`

4.1.64 CatpcaResult

`tabelz.analysis.categorical_pca.CatpcaResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`CatpcaConfig`	`required`
`n`	`int`	`required`
`missing_count`	`int`	`required`
`excluded_count`	`int`	`required`
`iterations`	`int`	`required`
`converged`	`bool`	`required`
`loss`	`float`	`required`
`stop_reason`	`str`	`''`
`dimensions`	`tuple[CatpcaDimension, ...]`	`required`
`variables`	`tuple[CatpcaVariable, ...]`	`required`
`categories`	`tuple[CatpcaCategory, ...]`	`required`
`artifact`	`CatpcaArtifact`	`required`
`notes`	`tuple[str, ...]`	`()`

4.1.65 CatpcaVariable

`tabelz.analysis.categorical_pca.CatpcaVariable`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`variable`	`str`	`required`
`level`	`str`	`required`
`loadings`	`tuple[float, ...]`	`required`
`categories`	`int`	`required`

4.1.66 ChaidArtifact

`tabelz.analysis.chaid.ChaidArtifact`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`target_kind`	`str`	`required`
`classes`	`tuple[str, ...]`	`required`
`predictors`	`tuple[str, ...]`	`required`
`numeric`	`tuple[str, ...]`	`required`
`cut_points`	`tuple[tuple[str, tuple[float, ...]], ...]`	`required`

Field	Type	Default
nodes	tuple[ChaidNode,...]	required
target	str	''
training_row_ids	tuple[str,...]	()
test_row_ids	tuple[str,...]	()

4.1.67 ChaidConfig

tabelz.analysis.chaid.ChaidConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
target	str	required
predictors	tuple[str,...]	required
target_kind	Literal['categorical','scale']	'categorical'
class_order	tuple[str,...]	()
ordinal	tuple[str,...]	()
bins	int	10
merge_alpha	float	0.05
split_alpha	float	0.05
min_parent	int	100
min_child	int	50
max_depth	int	3
exhaustive	bool	False
algorithm	Literal['merge_only','kass_resplit']	'merge_only'
resplit_alpha	float	0.01
partition_budget	int	100000
test_fraction	float	0.3
seed	int	20260928

predictors
MinLen(min_length=1); MaxLen(max_length=64)

class_order
MaxLen(max_length=200)

bins
Ge(ge=2); Le(le=100)

merge_alpha
Gt(gt=0.0); Lt(lt=1.0)

split_alpha
Gt(gt=0.0); Lt(lt=1.0)

min_parent
Ge(ge=2)

min_child
Ge(ge=1)
max_depth
Ge(ge=1); Le(le=10)
resplit_alpha
Gt(gt=0.0); Lt(lt=1.0)
partition_budget
Ge(ge=1); Le(le=1000000)
test_fraction
Ge(ge=0.1); Le(le=0.5)
seed
Ge(ge=0)

4.1.68 ChaidNode

`tabelz.analysis.chaid.ChaidNode`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`node`	`int`	`required`
`parent`	`int|None`	`required`
`depth`	`int`	`required`
`condition`	`str`	`required`
`n`	`int`	`required`
`distribution`	`tuple[int,...]`	`()`
`mean`	`float|None`	`None`
`prediction`	`str`	`required`
`split`	`ChaidSplitTest|None`	`None`
`children`	`tuple[int,...]`	`()`
`stop_reason`	`str`	`''`

4.1.69 ChaidResult

`tabelz.analysis.chaid.ChaidResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`ChaidConfig`	`required`
`training_rows`	`int`	`required`
`test_rows`	`int`	`required`
`missing_target`	`int`	`required`
`excluded_count`	`int`	`required`
`classes`	`tuple[str,...]`	`required`
`nodes`	`tuple[ChaidNode,...]`	`required`

Field	Type	Default
training_risk	float	required
test_risk	float	required
risk_name	str	required
test_table	tuple[tuple[int,...],...]	()
artifact	ChaidArtifact	required
notes	tuple[str,...]	()

4.1.70 ChaidSplitTest

tabelz.analysis.chaid.ChaidSplitTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
predictor	str	required
groups	tuple[tuple[str,...],...]	required
statistic	float|None	required
df	tuple[int,...]	required
p_value	float|None	required
multiplier	float	required
adjusted_p	float|None	required
eligible	bool	required
reason	str	''

4.1.71 ClassificationTreeCandidate

tabelz.analysis.classification_tree.ClassificationTreeCandidate

Common strict, immutable model baseline for public contracts.

Field	Type	Default
max_depth	int	required
min_samples_leaf	int	required
cv_log_loss	float	required

max_depth
Ge(ge=1); Le(le=8)

min_samples_leaf
Ge(ge=1)

cv_log_loss
Ge(ge=0)

4.1.72 ClassificationTreeConfig

`tabelz.analysis.classification_tree.ClassificationTreeConfig`

Strict replayable classification-tree settings.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	()
categorical_predictors	tuple[str,...]	()
test_fraction	float	0.25
cv_folds	int	5
max_depth	int	8
min_samples_leaf	int	1
seed	int	0
class_weight	Literal['none','balanced']	'none'
positive_class	str|None	None
split_assignment	SplitAssignment|None	None
max_rows	int	1000000
max_cells	int	20000000

test_fraction
Ge(ge=0.05); Le(le=0.5)
cv_folds
Ge(ge=2); Le(le=10)
max_depth
Ge(ge=1); Le(le=8)
min_samples_leaf
Ge(ge=1); Le(le=1000000)
seed
Ge(ge=0); Le(le=2147483647)
max_rows
Ge(ge=1); Le(le=1000000)
max_cells
Ge(ge=1); Le(le=20000000)

4.1.73 ClassificationTreePrediction

`tabelz.analysis.classification_tree.ClassificationTreePrediction`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
actual	str|None	None

Field	Type	Default
predicted	str	required
probabilities	tuple[float,...]	required

row_id
MinLen(min_length=1)

4.1.74 ClassificationTreeResult

tabelz.analysis.classification_tree.ClassificationTreeResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	required
categorical_predictors	tuple[str,...]	required
test_fraction	float	required
cv_folds	int	required
max_depth	int	required
min_samples_leaf	int	required
seed	int	required
class_weight	Literal['none','balanced']	required
positive_class	str|None	required
class_order	tuple[str,...]	required
split	SplitAssignment	required
fold_validation_row_ids	tuple[tuple[str,...],...]	required
selected_max_depth	int	required
selected_min_samples_leaf	int	required
selected_cv_log_loss	float	required
candidates	tuple[ClassificationTreeCandidate,...]	required
encoding	ClassificationTreeEncoding	required
artifact	ClassificationTreeArtifact	required
training_metrics	ClassificationTreeMetrics	required
test_metrics	ClassificationTreeMetrics	required
training_predictions	tuple[ClassificationTreePrediction,...]	required
test_predictions	tuple[ClassificationTreePrediction,...]	required

Field	Type	Default
source_count	int	required
excluded_count	int	required
missing_response_count	int	required
incomplete_predictor_count	int	required
unknown_category_count	int	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_response_count
Ge(ge=0)
incomplete_predictor_count
Ge(ge=0)
unknown_category_count
Ge(ge=0)

4.1.75 ClassificationTreeArtifact

tabelz.analysis.classification_tree_artifact.ClassificationTreeArtifact

Plain JSON tree nodes with per-leaf class probability vectors.

Field	Type	Default
feature_order	tuple[str,...]	required
class_order	tuple[str,...]	required
children_left	tuple[int,...]	required
children_right	tuple[int,...]	required
feature	tuple[int,...]	required
threshold	tuple[float,...]	required
class_probabilities	tuple[tuple[float,...],...]	required
n_node_samples	tuple[int,...]	required
encoding	ClassificationTreeEncoding	required

feature_order
MinLen(min_length=1); MaxLen(max_length=128)
class_order
MinLen(min_length=2); MaxLen(max_length=20)
children_left
MinLen(min_length=1); MaxLen(max_length=511)

children_right
MinLen(min_length=1); MaxLen(max_length=511)
feature
MinLen(min_length=1); MaxLen(max_length=511)
threshold
MinLen(min_length=1); MaxLen(max_length=511)
class_probabilities
MinLen(min_length=1); MaxLen(max_length=511)
n_node_samples
MinLen(min_length=1); MaxLen(max_length=511)

4.1.76 ClassificationTreeEncoding

`tabelz.analysis.classification_tree_encoding.ClassificationTreeEncoding`

Immutable schema learned from one training population.

Field	Type	Default
`numeric_predictors`	`tuple[str,...]`	`()`
`categorical_predictors`	`tuple[str,...]`	`()`
`categorical_levels`	`tuple[tuple[str,...],...]`	`()`
`feature_names`	`tuple[str,...]`	`required`

feature_names
MinLen(min_length=1); MaxLen(max_length=128)

4.1.77 ClassificationTreeMetrics

`tabelz.analysis.classification_tree_metrics.ClassificationTreeMetrics`

Metrics for one fixed training or untouched-test row set.

Field	Type	Default
`row_count`	`int`	`required`
`accuracy`	`float|None`	`None`
`log_loss`	`float|None`	`None`
`brier_score`	`float|None`	`None`
`confusion_matrix`	`tuple[tuple[int,...],...]`	`()`
`support`	`tuple[int,...]`	`()`
`precision`	`tuple[float|None,...]`	`()`
`recall`	`tuple[float|None,...]`	`()`
`roc_auc`	`float|None`	`None`
`average_precision`	`float|None`	`None`
`roc_curve`	`tuple[tuple[float,float,float|None],...]`	`()`

Field	Type	Default
pr_curve	tuple[tuple[float,float,float|None], ...]	()

row_count
Ge(ge=0)

4.1.78 KMeansConfig

tabelz.analysis.clustering.KMeansConfig

Frozen, strict k-means request: columns, k, seed, restarts, and limits.

Field	Type	Default
columns	tuple[str,...]	required
k	int	required
seed	int	0
restarts	int	10
standardize	bool	True
max_rows	int	1000000
max_cells	int	20000000

k
Ge(ge=2); Le(le=1000)
seed
Ge(ge=0)
restarts
Ge(ge=1); Le(le=100)
max_rows
Ge(ge=2); Le(le=1000000)
max_cells
Ge(ge=1); Le(le=20000000)

4.1.79 KMeansResult

tabelz.analysis.clustering.KMeansResult

Frozen, strict k-means outcome for transport and rendering.

Field	Type	Default
columns	tuple[str,...]	required
k	int	required
seed	int	required
restarts	int	required
standardize	bool	required
means	tuple[float,...]	required

Field	Type	Default
scales	tuple[float,...]	required
counts	tuple[int,...]	required
centroids	tuple[tuple[float,...],...]	required
inertia	float	required
iterations	int	required
cluster_labels	tuple[int,...]	required
points	tuple[tuple[float,float],...]	required
silhouette	float|None	None
silhouette_sample_size	int	0
row_ids	tuple[str,...]	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
warnings	tuple[str,...]	()

k
Ge(ge=2); Le(le=1000)
seed
Ge(ge=0)
restarts
Ge(ge=1); Le(le=100)
inertia
Ge(ge=0.0)
iterations
Ge(ge=1)
silhouette_sample_size
Ge(ge=0)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)

4.1.80 ContrastResult

`tabelz.analysis.contrast_families.ContrastResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	str	required

Field	Type	Default
variance	Literal['pooled','welch']	required
adjustment	str	required
alpha	float	required
rows	tuple[ContrastRow,...]	required
warnings	tuple[str,...]	()
notes	tuple[str,...]	()

4.1.81 ContrastRow

tabelz.analysis.contrast_families.ContrastRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
coefficients	tuple[float,...]	required
estimate	float	required
standard_error	float	required
df	float	required
t_value	float	required
p_value	float	required
p_adjusted	float|None	required
lower	float|None	required
upper	float|None	required

4.1.82 KendallConfig

tabelz.analysis.correlation_extensions.KendallConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
columns	tuple[str,...]	required
deletion	Literal['pairwise','listwise']	'pairwise'
inference	Literal['asymptotic','exact','permutation']	'asymptotic'
permutation_draws	int	10000
seed	int	20260928

columns
MinLen(min_length=2); MaxLen(max_length=12)

permutation_draws
Ge(ge=100); Le(le=200000)

seed
Ge(ge=0); Le(le=4294967295)

4.1.83 KendallPair

`tabelz.analysis.correlation_extensions.KendallPair`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	str	required
y	str	required
n	int	required
tau_b	float|None	required
s	int	required
z	float|None	required
p_value	float|None	required
note	str	''

4.1.84 KendallResult

`tabelz.analysis.correlation_extensions.KendallResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	KendallConfig	required
rows_analysed	int	required
excluded_count	int	required
pairs	tuple[KendallPair, ...]	required

4.1.85 PartialCorrelationConfig

`tabelz.analysis.correlation_extensions.PartialCorrelationConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	str	required
y	str	required
controls	tuple[str, ...]	required

controls
MinLen(min_length=1); MaxLen(max_length=10)

4.1.86 PartialCorrelationResult

`tabelz.analysis.correlation_extensions.PartialCorrelationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`PartialCorrelationConfig`	`required`
`n`	`int`	`required`
`excluded_count`	`int`	`required`
`incomplete_count`	`int`	`required`
`control_rank`	`int`	`required`
`zero_order_r`	`float`	`required`
`partial_r`	`float`	`required`
`df`	`int`	`required`
`t_value`	`float`	`required`
`p_value`	`float`	`required`
`note`	`str`	`required`

4.1.87 CorrespondenceCategory

`tabelz.analysis.correspondence_analysis.CorrespondenceCategory`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`label`	`str`	`required`
`mass`	`float`	`required`
`margin`	`int`	`required`
`coordinates`	`tuple[float|None, ...]`	`required`
`contributions`	`tuple[float|None, ...]`	`required`
`cosine_squared`	`tuple[float|None, ...]`	`required`
`active`	`bool`	`required`

4.1.88 CorrespondenceConfig

`tabelz.analysis.correspondence_analysis.CorrespondenceConfig`

Request for event-level categorical CA or a numeric count table.

Field	Type	Default
`mode`	`Literal['categorical','count_table']`	`'categorical'`
`row_column`	`str|None`	`None`
`column_column`	`str|None`	`None`
`frequency_column`	`str|None`	`None`

Field	Type	Default
row_label_column	str|None	None
count_columns	tuple[str,...]	()
dimensions	int	2
max_rows	int	1000000

dimensions
Ge(ge=1); Le(le=10)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.89 CorrespondenceResult

tabelz.analysis.correspondence_analysis.CorrespondenceResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
mode	Literal['categorical','count_table']	required
config	CorrespondenceConfig	required
source_count	int	required
eligible_count	int	required
missing_count	int	required
excluded_count	int	required
total	int	required
row_categories	tuple[CorrespondenceCategory,...]	required
column_categories	tuple[CorrespondenceCategory,...]	required
singular_values	tuple[float,...]	required
inertia	tuple[float,...]	required
total_inertia	float	required
chi_square	float	required
rank	int	required
requested_dimensions	int	required
dimensions	int	required
retained_row_ids	tuple[str,...]	()
warnings	tuple[str,...]	()
population_fingerprint	str	''
method	Literal['correspondence_analysis.simple_ca.v1']	'correspondence_analysis.simple_ca.v1'
implementation_version	Literal['1.0.0']	'1.0.0'

4.1.90 CoxRegressionCoefficient

`tabelz.analysis.cox_regression.CoxRegressionCoefficient`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`term`	`str`	`required`
`predictor`	`str`	`required`
`reference_level`	`str|None`	`None`
`log_hazard`	`float`	`required`
`standard_error`	`float`	`required`
`hazard_ratio`	`float|None`	`required`
`hazard_ratio_ci_lower`	`float|None`	`required`
`hazard_ratio_ci_upper`	`float|None`	`required`

4.1.91 CoxRegressionConfig

`tabelz.analysis.cox_regression.CoxRegressionConfig`

Frozen, bounded Cox PH request.

Field	Type	Default
`duration_column`	`str`	`required`
`event_column`	`str`	`required`
`event_value`	`str`	`required`
`censored_value`	`str`	`required`
`predictors`	`tuple[str,...]`	`required`
`categorical_predictors`	`tuple[str,...]`	`()`
`confidence_level`	`float`	`0.95`
`max_rows`	`int`	`1000000`
`max_iterations`	`int`	`200`
`max_diagnostic_rows`	`int`	`2000`

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

max_iterations
Ge(ge=1); Le(le=2000)

max_diagnostic_rows
Ge(ge=1); Le(le=10000)

4.1.92 CoxRegressionDiagnostic

`tabelz.analysis.cox_regression.CoxRegressionDiagnostic`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`status`	`Literal['computed','unavailable']`	required
`interpretation`	`str`	required
`by_term`	`tuple[dict[str,object],...]`	`()`
`residuals`	`tuple[SchoenfeldDiagnostic,...]`	`()`
`reason`	`str|None`	`None`

4.1.93 CoxRegressionResult

`tabelz.analysis.cox_regression.CoxRegressionResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`duration_column`	`str`	required
`event_column`	`str`	required
`event_value`	`str`	required
`censored_value`	`str`	required
`predictors`	`tuple[str,...]`	required
`categorical_predictors`	`tuple[str,...]`	required
`confidence_level`	`float`	required
`ties`	`Literal['efron']`	required
`max_iterations`	`int`	required
`max_rows`	`int`	required
`max_diagnostic_rows`	`int`	required
`coefficients`	`tuple[CoxRegressionCoefficient,...]`	required
`partial_log_likelihood`	`float`	required
`source_count`	`int`	required
`excluded_count`	`int`	required
`missing_count`	`int`	required
`analyzed_count`	`int`	required
`event_count`	`int`	required
`censored_count`	`int`	required
`design_rank`	`int`	required
`design_columns`	`tuple[str,...]`	required
`population_fingerprint`	`str`	required

Field	Type	Default
diagnostics	CoxRegressionDiagnostic	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
analyzed_count
Ge(ge=1)
event_count
Ge(ge=1)
censored_count
Ge(ge=0)
design_rank
Ge(ge=1)

4.1.94 SchoenfeldDiagnostic

tabelz.analysis.cox_regression.SchoenfeldDiagnostic

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
time	float	required
term	str	required
residual	float	required

4.1.95 CoxTdCoefficient

tabelz.analysis.cox_time_dependent_contracts.CoxTdCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
covariate	str	required
kind	Literal['covariate','time_function']	required
reference_level	str|None	None
coefficient	float	required
standard_error	float	required

Field	Type	Default
robust_standard_error	float|None	None
z	float	required
p_value	float	required
hazard_ratio	float|None	required
lower	float|None	required
upper	float|None	required

4.1.96 CoxTdPathSegment

tabelz.analysis.cox_time_dependent_contracts.CoxTdPathSegment

Common strict, immutable model baseline for public contracts.

Field	Type	Default
start	float	required
stop	float	required
values	tuple[tuple[str,str],...]	required

4.1.97 CoxTdPhRow

tabelz.analysis.cox_time_dependent_contracts.CoxTdPhRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
chisq	float	required
df	int	required
p_value	float	required

4.1.98 CoxTdPhTest

tabelz.analysis.cox_time_dependent_contracts.CoxTdPhTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
status	Literal['computed','withheld']	required
transform	Literal['km','rank','identity']	required
rows	tuple[CoxTdPhRow,...]	()
reason	str|None	None

4.1.99 CoxTdPrediction

`tabelz.analysis.cox_time_dependent_contracts.CoxTdPrediction`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`stratum`	`str|None`	`required`
`segments`	`tuple[CoxTdPathSegment,...]`	`required`
`rows`	`tuple[CoxTdPredictionRow,...]`	`required`

4.1.100 CoxTdPredictionRow

`tabelz.analysis.cox_time_dependent_contracts.CoxTdPredictionRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`time`	`float`	`required`
`cumulative_hazard`	`float`	`required`
`survival`	`float`	`required`

4.1.101 CoxTdSplit

`tabelz.analysis.cox_time_dependent_contracts.CoxTdSplit`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`function`	`Literal['none','time','log_time','step']`	`required`
`cut_count`	`int`	`required`
`rows_before`	`int`	`required`
`rows_after`	`int`	`required`
`exposure_before`	`float`	`required`
`exposure_after`	`float`	`required`

4.1.102 CoxTdTest

`tabelz.analysis.cox_time_dependent_contracts.CoxTdTest`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`name`	`str`	`required`
`chisq`	`float`	`required`

Field	Type	Default
df	int	required
p_value	float	required

4.1.103 CoxTimeDependentConfig

`tabelz.analysis.cox_time_dependent_contracts.CoxTimeDependentConfig`

A Cox model on (start, stop] episodes, or on one row per subject with x g(t) terms.

Field	Type	Default
layout	Literal['counting_process','one_row']	'one_row'
subject	str|None	None
start	str|None	None
stop	str	required
status	str	required
event_value	str	'1'
censored_value	str	'0'
numeric_covariates	tuple[str,...]	()
categorical_covariates	tuple[str,...]	()
time_function	Literal['none','time','log_time','step']	'none'
time_covariates	tuple[str,...]	()
time_origin	float	0.0
step_time	float|None	None
strata	str|None	None
ties	Literal['efron','breslow']	'efron'
robust	bool	False
ph_transform	Literal['km','rank','identity']	'km'
prediction_segments	str	''
confidence_level	float	0.95
max_split_rows	int	1000000
max_iterations	int	100

stop
MinLen(min_length=1)
status
MinLen(min_length=1)
event_value
MinLen(min_length=1)
censored_value
MinLen(min_length=1)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_split_rows
Ge(ge=1); Le(le=1000000)

max_iterations
Ge(ge=1); Le(le=500)

4.1.104 CoxTimeDependentResult

`tabelz.analysis.cox_time_dependent_contracts.CoxTimeDependentResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`CoxTimeDependentConfig`	`required`
`coefficients`	`tuple[CoxTdCoefficient,...]`	`required`
`tests`	`tuple[CoxTdTest,...]`	`required`
`log_likelihood`	`float`	`required`
`null_log_likelihood`	`float`	`required`
`iterations`	`int`	`required`
`source_rows`	`int`	`required`
`excluded_rows`	`int`	`required`
`missing_rows`	`int`	`required`
`subjects`	`int`	`required`
`missing_subjects`	`int`	`required`
`episodes`	`int`	`required`
`events`	`int`	`required`
`gaps`	`int`	`required`
`strata_levels`	`tuple[str,...]`	`required`
`split`	`CoxTdSplit`	`required`
`ph_test`	`CoxTdPhTest`	`required`
`prediction`	`CoxTdPrediction|None`	`None`
`notes`	`tuple[str,...]`	`()`
`warnings`	`tuple[str,...]`	`()`

4.1.105 CrossCorrelationConfig

`tabelz.analysis.cross_correlation.CrossCorrelationConfig`

Strict request for a bounded two-series cross-correlation.

Field	Type	Default
`series_x`	`str`	`required`
`series_y`	`str`	`required`
`time_column`	`str|None`	`None`

Field	Type	Default
calendar	Literal['strict','weekdays']	'strict'
max_lag	int	30
difference_order_x	int	0
difference_order_y	int	0
prewhiten	bool	False
prewhiten_order	int	1
max_rows	int	1000000
max_cells	int	20000000

max_lag
Ge(ge=0); Le(le=250)

difference_order_x
Ge(ge=0); Le(le=1)

difference_order_y
Ge(ge=0); Le(le=1)

prewhiten_order
Ge(ge=1); Le(le=20)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

4.1.106 CrossCorrelationLag

tabelz.analysis.cross_correlation.CrossCorrelationLag

One qualifying lag row: signed lag, Pearson correlation, overlap.

Field	Type	Default
lag	int	required
correlation	float	required
overlap	int	required
inside_band	bool	required

4.1.107 CrossCorrelationResult

tabelz.analysis.cross_correlation.CrossCorrelationResult

Bounded cross-correlation table, peak, and the exact input partition.

Field	Type	Default
status	Literal['ok','insufficient_data']	required
series_x	str	required
series_y	str	required

Field	Type	Default
time_column	str|None	required
calendar	Literal['strict','weekdays']	required
frequency	str	required
max_lag_used	int	required
band_basis	Literal['nominal','prewhitened']	required
prewhiten_order	int|None	required
difference_order_x	int	required
difference_order_y	int	required
n_valid_pairs	int	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
difference_missing_count	int	required
omitted_lag_count	int	required
peak_lag	int|None	required
peak_label	str|None	required
lag_rows	tuple[CrossCorrelationLag,...]	()
retained_row_ids	tuple[str,...]	()
population_fingerprint	str	required
warnings	tuple[str,...]	()
method	str	'cross_correlation.two_series'
implementation_version	str	'1.0.0'

4.1.108 CurveCoefficient

tabelz.analysis.curve_estimation.CurveCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
fitted_term	str	required
estimate	float	required
standard_error	float|None	required
t_value	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required

Field	Type	Default
parameter	float	required
parameter_lower	float|None	required
parameter_upper	float|None	required

4.1.109 CurveEstimationConfig

tabelz.analysis.curve_estimation.CurveEstimationConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
predictor	str	''
sequence	Literal['input_position','retained_order']|None	None
curves	tuple[Literal['linear','logarithmic','inverse','quadratic','cubic','power','compound','s','logistic','growth','exponential'],...]	required
upper_bound	float|None	None
confidence_level	float	0.95

curves
MinLen(min_length=1); MaxLen(max_length=11)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.110 CurveEstimationResult

tabelz.analysis.curve_estimation.CurveEstimationResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CurveEstimationConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
x_label	str	required
fits	tuple[CurveFit,...]	required
points_x	tuple[float,...]	()
points_y	tuple[float,...]	()
notes	tuple[str,...]	()

4.1.111 CurveFit

`tabelz.analysis.curve_estimation.CurveFit`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`curve`	`Literal['linear','logarithmic','inverse','quadratic','cubic','power','compound','s','logistic','growth','exponential']`	required
`equation`	`str`	required
`fitted_scale`	`str`	required
`available`	`bool`	required
`reason`	`str`	`''`
`r_squared`	`float|None`	None
`adjusted_r_squared`	`float|None`	None
`standard_error`	`float|None`	None
`f_value`	`float|None`	None
`df1`	`int|None`	None
`df2`	`int|None`	None
`p_value`	`float|None`	None
`coefficients`	`tuple[CurveCoefficient,...]`	()
`fitted_curve`	`Literal['mean','median']`	'mean'
`curve_x`	`tuple[float,...]`	()
`curve_y`	`tuple[float|None,...]`	()

4.1.112 DifferenceFinderOptions

`tabelz.analysis.difference_finder.DifferenceFinderOptions`

Versioned and inspectable rank policy for the Difference Finder.

Field	Type	Default
`version`	`Literal['1.0.0']`	'1.0.0'
`minimum_meaningful_difference`	`float`	0.0
`effect_weight`	`float`	1.0
`significance_weight`	`float`	1.0
`quantiles`	`tuple[float,...]`	(0.5,)
`multiplicity`	`Literal['benjamini_hochberg','none']`	'benjamini_hochberg'
`max_findings`	`int`	50
`paired`	`bool`	False
`entity_key`	`EntityKey|None`	None

minimum_meaningful_difference
Ge(ge=0.0)
effect_weight
Ge(ge=0.0)
significance_weight
Ge(ge=0.0)
max_findings
Ge(ge=1); Le(le=50)

4.1.113 DistributionDiagnosticsConfig

`tabelz.analysis.distribution_diagnostics.DistributionDiagnosticsConfig`

Strict request for one diagnostic on one numeric source column.

Field	Type	Default
`column`	`str`	`required`
`method`	`Literal['shapiro_wilk','anderson_darling','grubbs','poisson_gof']`	`'shapiro_wilk'`
`alpha`	`float`	`0.05`
`max_rows`	`int`	`1000000`
`max_plot_points`	`int`	`400`
`poisson_min_expected`	`float`	`5.0`

alpha
Gt(gt=0.0); Lt(lt=1.0)
max_rows
Ge(ge=1); Le(le=1000000)
max_plot_points
Ge(ge=1); Le(le=400)
poisson_min_expected
Gt(gt=0.0); Le(le=100.0)

4.1.114 DistributionDiagnosticsResult

`tabelz.analysis.distribution_diagnostics.DistributionDiagnosticsResult`

JSON-safe diagnostic result and bounded plot/table payload.

Field	Type	Default
`column`	`str`	`required`
`method_name`	`Literal['shapiro_wilk','anderson_darling','grubbs','poisson_gof']`	`required`
`alpha`	`float`	`required`
`confidence_level`	`float`	`required`
`statistic`	`float`	`required`

Field	Type	Default
p_value	float|None	required
critical_value	float|None	None
decision	str	required
source_count	int	required
observations	int	required
missing_count	int	required
excluded_count	int	required
retained_row_ids	tuple[str,...]	()
assumptions	tuple[str,...]	()
warnings	tuple[str,...]	()
candidate_row_id	str|None	None
candidate_value	float|None	None
candidate_tied_row_ids	tuple[str,...]	()
degrees_of_freedom	int|None	None
fitted_rate	float|None	None
bins	tuple[dict[str,object],...]	()
qq_points	tuple[dict[str,float],...]	()
probability_points	tuple[dict[str,float],...]	()
population_fingerprint	str|None	None
method	str	'distribution_diagnostics'
implementation_version	str	'1.0.0'

4.1.115 RandomDataResult

tabelz.analysis.distribution_sampling.RandomDataResult

A generated table without row identities plus its strict summary.

Field	Type	Default
frame	polars.dataframe.frame.DataFrame	required
summary	RandomDataSummary	required

4.1.116 BinomialDistribution

tabelz.analysis.distribution_utilities_contracts.BinomialDistribution

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['binomial']	required
trials	int	required
probability	float	required

4.1.117 ChiSquareDistribution

tabelz.analysis.distribution_utilities_contracts.ChiSquareDistribution

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['chi_square']	required
degrees_of_freedom	float	required

4.1.118 CurvePayload

tabelz.analysis.distribution_utilities_contracts.CurvePayload

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['continuous','discrete']	required
points	tuple[CurvePoint,...]	required
x_min	float	required
x_max	float	required
y_max	float	required
clipped	bool	required
stride	int	required
range_probability	float	required
shade_lower	float|None	required
shade_upper	float|None	required
shade_probability	float|None	required

4.1.119 CurvePoint

tabelz.analysis.distribution_utilities_contracts.CurvePoint

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	float	required
value	float	required
cdf	float	required

4.1.120 DistributionSummary

`tabelz.analysis.distribution_utilities_contracts.DistributionSummary`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['normal','t','chi_square','f','binomial','poisson','exponential','weibull']	required
label	str	required
kind	Literal['continuous','discrete']	required
parameters	tuple[NamedParameter,...]	required
support_lower	float|None	required
support_upper	float|None	required
mean	float|None	required
variance	float|None	required
standard_deviation	float|None	required

4.1.121 ExponentialDistribution

`tabelz.analysis.distribution_utilities_contracts.ExponentialDistribution`

Exponential distribution parameterized by its scale, which equals its mean.

Field	Type	Default
family	Literal['exponential']	required
scale	float	required

4.1.122 FDistribution

`tabelz.analysis.distribution_utilities_contracts.FDistribution`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['f']	required
numerator_df	float	required
denominator_df	float	required

4.1.123 IntervalRequest

`tabelz.analysis.distribution_utilities_contracts.IntervalRequest`

An interval with optional bounds; “None“ means unbounded on that side.

Field	Type	Default
lower	float|None	None
upper	float|None	None
lower_inclusive	bool	True
upper_inclusive	bool	True

4.1.124 IntervalResult

tabelz.analysis.distribution_utilities_contracts.IntervalResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
lower	float|None	required
upper	float|None	required
lower_inclusive	bool	required
upper_inclusive	bool	required
integer_lower	int|None	required
integer_upper	int|None	required
probability	float	required
complement	float	required
log10_probability	float|None	required
endpoint_rule	str	required

4.1.125 NamedParameter

tabelz.analysis.distribution_utilities_contracts.NamedParameter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
value	float	required

4.1.126 NormalDistribution

tabelz.analysis.distribution_utilities_contracts.NormalDistribution

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['normal']	required
mean	float	0.0
standard_deviation	float	1.0

4.1.127 PoissonDistribution

`tabelz.analysis.distribution_utilities_contracts.PoissonDistribution`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`family`	`Literal['poisson']`	required
`mean`	`float`	required

4.1.128 ProbabilityCalculationConfig

`tabelz.analysis.distribution_utilities_contracts.ProbabilityCalculationConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`distribution`	`NormalDistribution|StudentTDistribution|ChiSquareDistribution|FDistribution|BinomialDistribution|PoissonDistribution|ExponentialDistribution|WeibullDistribution`	required
`x_values`	`tuple[float, ...]`	`()`
`probabilities`	`tuple[float, ...]`	`()`
`interval`	`IntervalRequest|None`	`None`
`curve_points`	`int`	`201`

x_values
MaxLen(max_length=200)

probabilities
MaxLen(max_length=200)

curve_points
Ge(ge=21); Le(le=512)

4.1.129 ProbabilityCalculationResult

`tabelz.analysis.distribution_utilities_contracts.ProbabilityCalculationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`method_identity`	`str`	required
`implementation_version`	`str`	required
`scipy_version`	`str`	required
`distribution`	`DistributionSummary`	required
`values`	`tuple[ValueRow, ...]`	required

Field	Type	Default
quantiles	tuple[QuantileRow,...]	required
interval	IntervalResult|None	required
curve	CurvePayload	required
warnings	tuple[str,...]	required
disclosures	tuple[str,...]	required

4.1.130 QuantileRow

tabelz.analysis.distribution_utilities_contracts.QuantileRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
probability	float	required
lower_quantile	float|None	required
upper_quantile	float|None	required
achieved_cdf	float|None	required
achieved_sf	float|None	required

4.1.131 RandomColumnSummary

tabelz.analysis.distribution_utilities_contracts.RandomColumnSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
count	int	required
mean	float|None	required
standard_deviation	float|None	required
minimum	float	required
maximum	float	required

4.1.132 RandomDataConfig

tabelz.analysis.distribution_utilities_contracts.RandomDataConfig

A bounded seeded table request: one distribution, independent columns.

Field	Type	Default
distribution	NormalDistribution|StudentTDistribution|ChiSquareDistribution|FDistribution|BinomialDistribution|PoissonDistribution|ExponentialDistribution|WeibullDistribution	required
rows	int	required
column_names	tuple[str,...]	required
seed	int	required
algorithm	Literal['pcg64-inverse-cdf-v1']	'pcg64-inverse-cdf-v1'

rows
Ge(ge=1); Le(le=1000000)

column_names
MinLen(min_length=1); MaxLen(max_length=20)

seed
Ge(ge=0); Le(le=9223372036854775807)

4.1.133 RandomDataSummary

tabelz.analysis.distribution_utilities_contracts.RandomDataSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method_identity	str	required
implementation_version	str	required
algorithm	str	required
algorithm_description	str	required
numpy_version	str	required
scipy_version	str	required
seed	int	required
rows	int	required
column_names	tuple[str,...]	required
dtype	Literal['float64','int64']	required
distribution	DistributionSummary	required
columns	tuple[RandomColumnSummary,...]	required
preview_rows	tuple[tuple[float,...],...]	required
output_fingerprint	str	required

4.1.134 StudentTDistribution

tabelz.analysis.distribution_utilities_contracts.StudentTDistribution

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['t']	required
degrees_of_freedom	float	required

4.1.135 ValueRow

tabelz.analysis.distribution_utilities_contracts.ValueRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	float	required
density	float|None	required
mass	float|None	required
cdf	float	required
sf	float	required
log10_cdf	float|None	required
log10_sf	float|None	required
in_support	bool	required

4.1.136 WeibullDistribution

tabelz.analysis.distribution_utilities_contracts.WeibullDistribution

Two-parameter Weibull distribution with support starting at zero.

Field	Type	Default
family	Literal['weibull']	required
shape	float	required
scale	float	required

4.1.137 DoeAnalysisResult

tabelz.analysis.doe.DoeAnalysisResult

Field	Type	Default
response	str	required
factors	tuple[str,...]	required
observations	int	required
dropped_rows	int	required
terms	tuple[DoeTermResult,...]	required
anova	tuple[DoeAnovaRow,...]	required

Field	Type	Default
r_squared	float	required
adjusted_r_squared	float|None	required
rmse	float|None	required
residual_df	int	required
confidence_level	float	required
critical_t	float|None	required
actual_predicted	tuple[tuple[float,float],...]	required
residuals	tuple[tuple[float,float],...]	required
qq	tuple[tuple[float,float],...]	required
maximize_settings	tuple[tuple[str,float],...]	required
maximize_prediction	float	required
minimize_settings	tuple[tuple[str,float],...]	required
minimize_prediction	float	required
warnings	tuple[str,...]	()

4.1.138 DoeAnovaRow

tabelz.analysis.doe.DoeAnovaRow

Field	Type	Default
source	str	required
df	int	required
sum_squares	float	required
mean_square	float|None	required
f_value	float|None	required
p_value	float|None	required

4.1.139 DoeTermResult

tabelz.analysis.doe.DoeTermResult

Field	Type	Default
term	str	required
coefficient	float	required
effect	float	required
std_error	float|None	required
t_value	float|None	required
p_value	float|None	required
lower_ci	float|None	required
upper_ci	float|None	required
sum_squares	float|None	required

4.1.140 FactorDefinition

`tabelz.analysis.doe.FactorDefinition`

Field	Type	Default
`name`	`str`	`required`
`low`	`float`	`required`
`high`	`float`	`required`

4.1.141 FactorialDesignConfig

`tabelz.analysis.doe.FactorialDesignConfig`

Field	Type	Default
`factors`	`tuple[FactorDefinition,...]`	`required`
`response_name`	`str`	`'Response'`
`replicates`	`int`	`1`
`center_points`	`int`	`0`
`randomize`	`bool`	`True`
`seed`	`int`	`2026`

4.1.142 DoePowerConfig

`tabelz.analysis.doe_power.DoePowerConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`family`	`Literal['full_factorial','fractional_factorial','plackett_burman']`	`'full_factorial'`
`factors`	`int`	`3`
`resolution`	`int`	`4`
`replicates`	`int`	`1`
`center_points`	`int`	`0`
`blocks`	`int`	`1`
`block_seed`	`int`	`2026`
`model`	`Literal['full','main_2fi','main']`	`'full'`
`omitted_terms`	`tuple[str,...]`	`()`
`curvature_term`	`bool`	`True`
`effect_term`	`str`	`'A'`
`delta`	`float`	`1.0`
`sigma`	`float`	`1.0`
`alpha`	`float`	`0.05`
`target_power`	`float`	`0.8`

Field	Type	Default
solve_for	Literal['power','replicates','center_points','curve']	'power'
design	DoePowerDesign|None	None

factors
Ge(ge=2); Le(le=11)
resolution
Ge(ge=3); Le(le=8)
replicates
Ge(ge=1); Le(le=50)
center_points
Ge(ge=0); Le(le=50)
blocks
Ge(ge=1); Le(le=8)
block_seed
Ge(ge=0); Le(le=2147483647)
delta
Gt(gt=0.0)
sigma
Gt(gt=0.0)
alpha
Gt(gt=0.0); Lt(lt=0.5)
target_power
Gt(gt=0.0); Lt(lt=1.0)

4.1.143 DoePowerDesign

tabelz.analysis.doe_power.DoePowerDesign

The created design table a power calculation is tied to, and how its factors are coded.

Field	Type	Default
table_id	str	required
revision	int	required
table_title	str	''
factor_names	tuple[str,...]	required
factor_lows	tuple[float,...]	required
factor_highs	tuple[float,...]	required
foldover	bool	False

table_id
MinLen(min_length=1)
revision
Ge(ge=0)

4.1.144 DoePowerResult

`tabelz.analysis.doe_power.DoePowerResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	DoePowerConfig	required
solved_replicates	int	required
solved_center_poin ts	int	required
total_runs	int	required
corner_runs	int	required
rank	int	required
error_df	int	required
model_terms	int	required
noncentrality	float	required
critical_t	float	required
power	float	required
target_met	bool	required
unattainable	str	''
terms	tuple[TermPower,...]	required
curve	tuple[CurvePoint,...]	()
model_label	str	required
notes	tuple[str,...]	()
design_basis	Literal['standalone','design_table']	'standalone'
calculator_only	bool	False
matches_catalog	bool|None	None
excluded_rows	int	0
design_statement	str	''

4.1.145 TermPower

`tabelz.analysis.doe_power.TermPower`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
aliases	str	required
variance_factor	float|None	required
noncentrality	float|None	required
power	float|None	required
note	str	''

4.1.146 DoeModelArtifact

`tabelz.analysis.doe_stored_model.DoeModelArtifact`

An immutable DOE fit: coding, terms, coefficients, covariance and residual df.

Field	Type	Default
method	Literal['tabelz.analysis.doe_stored_model']	'tabelz.analysis.doe_stored_model'
schema_version	Literal['1.0.0','1.1.0','1.2.0','1.3.0']	'1.2.0'
response	str	required
design_family	str	required
fit_method	str	required
split_plot	bool	required
factors	tuple[DoeModelFactor,...]	required
terms	tuple[DoeModelTerm,...]	required
coefficients	tuple[float,...]	required
covariance	tuple[tuple[float,...],...]|None	required
residual_df	int	required
residual_variance	float|None	required
observations	int	required
fitted_ranges	dict[str,tuple[float,float]]	factory:dict
reference_block	str|None	None
split_plot_inference	Literal['kenward_roger','wald']|None	None
kenward_roger	KenwardRogerState|None	None
block_levels	tuple[str,...]	()
block_prediction	Literal['reference','average']	'reference'
feasible_constraint	str	''

response
MinLen(min_length=1)
factors
MinLen(min_length=1)
terms
MinLen(min_length=1)
residual_df
Ge(ge=0)
observations
Ge(ge=1)

4.1.147 DoeModelFactor

`tabelz.analysis.doe_stored_model.DoeModelFactor`

A design factor: coded from its natural “low“/“high“ settings, or categorical.

Field	Type	Default
name	str	required
kind	Literal['coded','categorical','mixture']	required
low	float|None	None
high	float|None	None
levels	tuple[str,...]	()

name
MinLen(min_length=1)

4.1.148 DoeModelTerm

tabelz.analysis.doe_stored_model.DoeModelTerm

One model column: the product of coded factor powers and level indicators.

Field	Type	Default
name	str	required
powers	tuple[tuple[str,int],...]	()
indicators	tuple[tuple[str,str],...]	()
block	str|None	None

name
MinLen(min_length=1)

4.1.149 KenwardRogerState

tabelz.analysis.doe_stored_model.KenwardRogerState

The Kenward-Roger state of a split-plot fit: the plug-in covariance Phi, the P_i terms and W, from which any contrast's df and scaling follow (the adjusted covariance is the artifact's covariance).

Field	Type	Default
phi	tuple[tuple[float,...],...]	required
p_terms	tuple[tuple[tuple[float,...],...],...]	required
w	tuple[tuple[float,...],...]	required

4.1.150 AnovaRow

tabelz.analysis.doe_taguchi_analysis.AnovaRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
source	str	required
df	int	required
ss	float	required
ms	float|None	required
f	float|None	required
p	float|None	required

4.1.151 Prediction

tabelz.analysis.doe_taguchi_analysis.Prediction

Common strict, immutable model baseline for public contracts.

Field	Type	Default
summary	Literal['sn','mean','sd','log_sd']	required
value	float|None	required
note	str	''

4.1.152 ResponseTable

tabelz.analysis.doe_taguchi_analysis.ResponseTable

Common strict, immutable model baseline for public contracts.

Field	Type	Default
summary	Literal['sn','mean','sd','log_sd']	required
levels	tuple[str,...]	required
averages	tuple[tuple[float|None,...],...]	required
deltas	tuple[float|None,...]	required
ranks	tuple[int|None,...]	required
unavailable	str	''

4.1.153 RunSummary

tabelz.analysis.doe_taguchi_analysis.RunSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
run	str	required
levels	tuple[str,...]	required
count	int	required
mean	float|None	required

Field	Type	Default
sd	float|None	required
log_sd	float|None	required
sn	float|None	required
slope	float|None	None
note	str	''

4.1.154 TaguchiAnalysisConfig

tabelz.analysis.doe_taguchi_analysis.TaguchiAnalysisConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factors	tuple[str,...]	required
responses	tuple[str,...]	required
sn	Literal['larger','smaller','nominal_variance','nominal_mean_sd','dynamic']	required
signal_values	tuple[float,...]	()
dynamic_reference	Literal['intercept','origin']	'intercept'
predict_terms	tuple[str,...]	()
predict_levels	tuple[str,...]	()
anova_summary	Literal['none','sn','mean','sd']	'none'
pooled	tuple[str,...]	()
seed	int	0

seed
Ge(ge=0)

4.1.155 TaguchiAnalysisResult

tabelz.analysis.doe_taguchi_analysis.TaguchiAnalysisResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	TaguchiAnalysisConfig	required
runs	tuple[RunSummary,...]	required
tables	tuple[ResponseTable,...]	required
predictions	tuple[Prediction,...]	required
anova	tuple[AnovaRow,...]	()
anova_note	str	''
warnings	tuple[str,...]	()

4.1.156 AssignmentRow

`tabelz.analysis.doe_taguchi_design.AssignmentRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`factor`	`str`	`required`
`column`	`int`	`required`
`levels`	`int`	`required`

4.1.157 InteractionRow

`tabelz.analysis.doe_taguchi_design.InteractionRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`interaction`	`str`	`required`
`degrees_of_freedom`	`int`	`required`
`estimable_df`	`int`	`required`
`correlated_with`	`str`	`required`
`note`	`str`	`required`

4.1.158 TaguchiDesign

`tabelz.analysis.doe_taguchi_design.TaguchiDesign`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`TaguchiDesignConfig`	`required`
`array_label`	`str`	`required`
`provenance`	`str`	`required`
`runs`	`int`	`required`
`assignment`	`tuple[AssignmentRow,...]`	`required`
`unused_columns`	`tuple[int,...]`	`required`
`interactions`	`tuple[InteractionRow,...]`	`required`
`outer_runs`	`tuple[tuple[int,...],...]`	`required`
`response_columns`	`tuple[str,...]`	`required`
`signal_by_column`	`tuple[float|None,...]`	`required`
`columns`	`tuple[str,...]`	`required`
`rows`	`tuple[tuple[int,...],...]`	`required`

4.1.159 TaguchiDesignConfig

`tabelz.analysis.doe_taguchi_design.TaguchiDesignConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`array`	`Literal['L4','L8','L9','L12','L16','L16_4','L18','L25','L27','L32','L36']`	`'L8'`
`factors`	`tuple[str,...]`	`('A','B','C')`
`columns`	`tuple[int,...]`	`()`
`interactions`	`tuple[str,...]`	`()`
`noise`	`Literal['repeats','outer']`	`'repeats'`
`repeats`	`int`	`2`
`outer_array`	`Literal['L4','L8','L9','L12','L16','L16_4','L18','L25','L27','L32','L36']`	`'L4'`
`noise_factors`	`tuple[str,...]`	`()`
`signal_levels`	`tuple[float,...]`	`()`
`randomize`	`bool`	`True`
`seed`	`int`	`0`

repeats
Ge(ge=1); Le(le=20)
seed
Ge(ge=0); Le(le=2147483647)

4.1.160 EquivalenceConfig

`tabelz.analysis.equivalence.EquivalenceConfig`

Replayable request for a Welch two-sample TOST.

Field	Type	Default
`response_column`	`str`	`required`
`group_column`	`str`	`required`
`group_a`	`str`	`required`
`group_b`	`str`	`required`
`lower_bound`	`float`	`required`
`upper_bound`	`float`	`required`
`alpha`	`float`	`0.05`
`max_rows`	`int`	`1000000`
`max_cells`	`int`	`20000000`

response_column
MinLen(min_length=1)
group_column
MinLen(min_length=1)

group_a
MinLen(min_length=1)

group_b
MinLen(min_length=1)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

4.1.161 EquivalenceResult

`tabelz.analysis.equivalence.EquivalenceResult`

Complete, bounded Welch TOST result and population accounting.

Field	Type	Default
`method`	`str`	`'WelchTOST(independenttwo-samplemeans)'`
`response_column`	`str`	required
`group_column`	`str`	required
`group_a`	`str`	required
`group_b`	`str`	required
`n_a`	`int`	required
`n_b`	`int`	required
`mean_a`	`float`	required
`mean_b`	`float`	required
`sd_a`	`float`	required
`sd_b`	`float`	required
`difference_a_minus_b`	`float`	required
`standard_error`	`float`	required
`degrees_of_freedom`	`float`	required
`lower_statistic`	`float`	required
`upper_statistic`	`float`	required
`p_lower`	`float`	required
`p_upper`	`float`	required
`confidence_level`	`float`	required
`interval_lower`	`float`	required
`interval_upper`	`float`	required
`lower_bound`	`float`	required
`upper_bound`	`float`	required
`equivalent`	`bool`	required
`decision`	`Literal['equivalence_established','equivalence_not_established']`	required
`source_count`	`int`	required

Field	Type	Default
excluded_count	int	required
missing_count	int	required
outside_group_count	int	required
analyzed_count	int	required
row_ids_a	tuple[str,...]	required
row_ids_b	tuple[str,...]	required
population_fingerprint	str	required
warnings	tuple[str,...]	()

n_a
Ge(ge=2)
n_b
Ge(ge=2)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
outside_group_count
Ge(ge=0)
analyzed_count
Ge(ge=4)
population_fingerprint
MinLen(min_length=64); MaxLen(max_length=64)

4.1.162 EquivalenceExtensionsConfig

tabelz.analysis.equivalence_extensions.EquivalenceExtensionsConfig

Replayable request for one-sample or paired equivalence.

Field	Type	Default
mode	Literal['one_sample_difference','paired_difference','paired_log_ratio']	'one_sample_difference'
response_column	str|None	None
first_column	str|None	None
second_column	str|None	None
target	float|None	None
lower_bound	float	required
upper_bound	float	required
alpha	float	0.05

Field	Type	Default
max_rows	int	1000000
max_cells	int	20000000

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

4.1.163 EquivalenceExtensionsResult

tabelz.analysis.equivalence_extensions.EquivalenceExtensionsResult

Complete bounded result with explicit display scale and row accounting.

Field	Type	Default
mode	Literal['one_sample_difference','paired_difference','paired_log_ratio']	required
response_column	str|None	required
first_column	str|None	required
second_column	str|None	required
target	float|None	required
lower_bound	float	required
upper_bound	float	required
alpha	float	required
confidence_level	float	required
sample_size	int	required
estimate	float	required
standard_error	float	required
degrees_of_freedom	float	required
p_lower	float	required
p_upper	float	required
interval_lower	float	required
interval_upper	float	required
transformed_estimate	float|None	required
transformed_interval_lower	float|None	required
transformed_interval_upper	float|None	required
display_scale	Literal['difference','ratio']	required
equivalent	bool	required
decision	Literal['equivalence_established','equivalence_not_established']	required

Field	Type	Default
source_count	int	required
excluded_count	int	required
missing_count	int	required
analyzed_count	int	required
row_ids	tuple[str,...]	required
population_fingerprint	str	required
warnings	tuple[str,...]	()
method_identity	str	'equivalence.one_sample_paired_tost'
implementation_version	str	'1.0.0'

sample_size
Ge(ge=2)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
analyzed_count
Ge(ge=2)
population_fingerprint
MinLen(min_length=64); MaxLen(max_length=64)

4.1.164 BarData

tabelz.analysis.exploration_extensions_contracts.BarData

Grouped or stacked counted/summarized bars over all retained rows.

Field	Type	Default
group_column	str	required
segment_column	str|None	required
aggregation	Literal['count','mean','median','sum']	required
denominator	Literal['none','row','column','overall']	required
stacked	bool	required
groups	tuple[BarGroup,...]	()
rendered_count	int	required
disclosure	str	required

Field	Type	Default
equivalent_table	EquivalentTable	required

4.1.165 BarGroup

tabelz.analysis.exploration_extensions_contracts.BarGroup

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
source_count	int	required
contributing_row_count	int	required
segments	tuple[BarSegment, ...]	()

4.1.166 BarSegment

tabelz.analysis.exploration_extensions_contracts.BarSegment

One series mark inside a group; aggregated, never one source row.

Field	Type	Default
label	str	required
value	float	required
valid_n	int	required
source_count	int	required
contributing_row_count	int	required
denominator_total	int|None	None
percentage	float|None	None

4.1.167 EcdfData

tabelz.analysis.exploration_extensions_contracts.EcdfData

Common strict, immutable model baseline for public contracts.

Field	Type	Default
value_column	str	required
group_column	str|None	required
proportion	bool	required
series	tuple[EcdfSeries, ...]	()
rendered_count	int	required
identity_preserved	bool	required

Field	Type	Default
disclosure	str	required
equivalent_table	EquivalentTable	required

4.1.168 EcdfPoint

tabelz.analysis.exploration_extensions_contracts.EcdfPoint

One exact ECDF step; tied values form a single step of full height.

Field	Type	Default
value	float	required
cumulative	float	required
tie_count	int	required
row_id	str|None	None

4.1.169 EcdfSeries

tabelz.analysis.exploration_extensions_contracts.EcdfSeries

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
missing	bool	required
source_count	int	required
rendered_count	int	required
identity_preserved	bool	required
points	tuple[EcdfPoint, ...]	()

4.1.170 EquivalentTable

tabelz.analysis.exploration_extensions_contracts.EquivalentTable

The exact aggregation over all retained rows, independent of display bounds.

Field	Type	Default
columns	tuple[str, ...]	required
rows	tuple[tuple[str|int|float|None, ...], ...]	()

4.1.171 ExplorationGraphConfig

tabelz.analysis.exploration_extensions_contracts.ExplorationGraphConfig

Strict request for one categorical bar, two-way heatmap, or ECDF graph.

Field	Type	Default
graph_kind	Literal['bar','heatmap','ecdf','pie']	'bar'
value_column	str|None	None
group_columns	tuple[str,...]	()
aggregation	Literal['count','mean','median','sum']	'count'
denominator	Literal['none','row','column','overall']	'none'
stacked	bool	False
row_column	str|None	None
column_column	str|None	None
group_column	str|None	None
proportion	bool	True
ordering	Literal['alphabetical','declared','frequency']	'alphabetical'
missing_groups	Literal['include','omit']	'include'
max_rows	int	1000000
max_cells	int	2000000

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=2000000)

4.1.172 ExplorationGraphResult

tabelz.analysis.exploration_extensions_contracts.ExplorationGraphResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
graph_kind	Literal['bar','heatmap','ecdf','pie']	required
value_column	str|None	required
group_columns	tuple[str,...]	required
row_column	str|None	required
column_column	str|None	required
group_column	str|None	required
aggregation	Literal['count','mean','median','sum']	required
denominator	Literal['none','row','column','overall']	required
stacked	bool	required
proportion	bool	required

Field	Type	Default
ordering	Literal['alphabetical','declared','frequency']	required
missing_groups	Literal['include','omit']	required
source_count	int	required
observations	int	required
valid_n	int	required
missing_count	int	required
excluded_count	int	required
bars	BarData|None	None
pie	PieData|None	None
heatmap	HeatmapData|None	None
ecdf	EcdfData|None	None
warnings	tuple[str,...]	()
retained_row_ids	tuple[str,...]	()
population_fingerprint	str	required
method	str	'exploration_extensions.graphs'
implementation_version	str	'1.1.0'

4.1.173 HeatmapCell

tabelz.analysis.exploration_extensions_contracts.HeatmapCell

One aggregated two-way cell; it cannot identify a single source row.

Field	Type	Default
row_label	str	required
column_label	str	required
value	float|None	required
aggregation	Literal['count','mean','median']	required
source_count	int	required
contributing_row_count	int	required

4.1.174 HeatmapData

tabelz.analysis.exploration_extensions_contracts.HeatmapData

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_column	str	required
column_column	str	required
value_column	str|None	required
aggregation	Literal['count','mean','median']	required
row_labels	tuple[str,...]	()
column_labels	tuple[str,...]	()
cells	tuple[HeatmapCell,...]	()
value_minimum	float|None	None
value_maximum	float|None	None
rendered_count	int	required
disclosure	str	required
equivalent_table	EquivalentTable	required

4.1.175 PieData

tabelz.analysis.exploration_extensions_contracts.PieData

Common strict, immutable model baseline for public contracts.

Field	Type	Default
group_column	str	required
slices	tuple[PieSlice,...]	()
rendered_count	int	required
disclosure	str	required
equivalent_table	EquivalentTable	required

4.1.176 PieSlice

tabelz.analysis.exploration_extensions_contracts.PieSlice

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
count	int	required
percentage	float	required

count
Ge(ge=0)

percentage
Ge(ge=0); Le(le=100)

4.1.177 ContourConfig

`tabelz.analysis.exploration_graph_contracts.ContourConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	str	required
y	str	required
z	str	required
view	Literal['contour','surface']	'contour'
aggregation	Literal['refuse','mean']	'refuse'
grid	int	51
levels	int	10
elevation	float	30.0
azimuth	float	45.0

grid
Ge(ge=11); Le(le=101)
levels
Ge(ge=3); Le(le=20)
elevation
Ge(ge=-90.0); Le(le=90.0)
azimuth
Ge(ge=0.0); Le(le=360.0)

4.1.178 ContourResult

`tabelz.analysis.exploration_graph_contracts.ContourResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ContourConfig	required
source_count	int	required
excluded_count	int	required
incomplete_count	int	required
locations	int	required
duplicates_collapsed	int	required
aggregated_locations	int	required
aggregated_rows	int	required
x_nodes	tuple[float,...]	required
y_nodes	tuple[float,...]	required
z_grid	tuple[tuple[float|None,...],...]	required

Field	Type	Default
levels	tuple[float,...]	required
z_range	tuple[float,float]	required
outside_hull_nodes	int	required
method	str	required

4.1.179 MatrixHistogram

tabelz.analysis.exploration_graph_contracts.MatrixHistogram

Common strict, immutable model baseline for public contracts.

Field	Type	Default
edges	tuple[float,...]	required
counts	tuple[int,...]	required

4.1.180 MatrixPlotConfig

tabelz.analysis.exploration_graph_contracts.MatrixPlotConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
columns	tuple[str,...]	required
color	str|None	None
diagonal	Literal['histogram','names']	'histogram'

columns
MinLen(min_length=2); MaxLen(max_length=8)

4.1.181 MatrixPlotResult

tabelz.analysis.exploration_graph_contracts.MatrixPlotResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MatrixPlotConfig	required
source_count	int	required
excluded_count	int	required
complete_count	int	required
incomplete_count	int	required
sampled	bool	required
sample_seed	int	required
row_ids	tuple[str,...]	required

Field	Type	Default
values	tuple[tuple[float,...],...]	required
groups	tuple[str,...]	()
group_levels	tuple[str,...]	()
ranges	tuple[tuple[float,float],...]	required
histograms	tuple[MatrixHistogram,...]	required

4.1.182 ProbabilityFit

tabelz.analysis.exploration_graph_contracts.ProbabilityFit

Common strict, immutable model baseline for public contracts.

Field	Type	Default
n	int	required
location	float	required
scale	float	required
parameters	tuple[ProbabilityParameter,...]	required
log_likelihood	float	required
anderson_darling	float	required
line	tuple[tuple[float,float,float],...]	required
band_lower	tuple[tuple[float,float,float],...]	()
band_upper	tuple[tuple[float,float,float],...]	()
band_reason	str	''
percentiles	tuple[ProbabilityPercentile,...]	()

4.1.183 ProbabilityGroup

tabelz.analysis.exploration_graph_contracts.ProbabilityGroup

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
n	int	required
tied_values	int	required
points	tuple[tuple[float,float,float],...]	required
plotted_count	int	required
fit	ProbabilityFit|None	None
fit_reason	str	''

4.1.184 ProbabilityParameter

tabelz.analysis.exploration_graph_contracts.ProbabilityParameter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
standard_error	float|None	required

4.1.185 ProbabilityPercentile

tabelz.analysis.exploration_graph_contracts.ProbabilityPercentile

Common strict, immutable model baseline for public contracts.

Field	Type	Default
percent	float	required
estimate	float	required
lower	float|None	required
upper	float|None	required

4.1.186 ProbabilityPlotConfig

tabelz.analysis.exploration_graph_contracts.ProbabilityPlotConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
family	Literal['normal','lognormal','weibull','exponential','loglogistic','sev']	'normal'
origin	float	0.0
group	str|None	None
pooled	bool	False
confidence	float	0.95

confidence
Gt(gt=0.5); Lt(lt=1.0)

4.1.187 ProbabilityPlotResult

tabelz.analysis.exploration_graph_contracts.ProbabilityPlotResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ProbabilityPlotConfig	required
excluded_count	int	required

Field	Type	Default
missing_count	int	required
outside_support_count	int	required
log_scale	bool	required
groups	tuple[ProbabilityGroup,...]	required
pooled_fit	ProbabilityFit|None	None
pooled_reason	str	''
probability_range	tuple[float,float]	required
probability_ticks	tuple[tuple[float,float],...]	required

4.1.188 Scatter3dConfig

tabelz.analysis.exploration_graph_contracts.Scatter3dConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	str	required
y	str	required
z	str	required
color	str|None	None
elevation	float	30.0
azimuth	float	45.0

elevation
Ge(ge=-90.0); Le(le=90.0)

azimuth
Ge(ge=0.0); Le(le=360.0)

4.1.189 Scatter3dResult

tabelz.analysis.exploration_graph_contracts.Scatter3dResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	Scatter3dConfig	required
source_count	int	required
excluded_count	int	required
complete_count	int	required
sampled	bool	required
sample_seed	int	required
row_ids	tuple[str,...]	required

Field	Type	Default
points	tuple[tuple[float,float,float],...]	required
groups	tuple[str,...]	()
group_levels	tuple[str,...]	()
ranges	tuple[tuple[float,float],...]	required

4.1.190 StemLeafConfig

tabelz.analysis.exploration_graph_contracts.StemLeafConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
leaf_unit	float|None	None

4.1.191 StemLeafResult

tabelz.analysis.exploration_graph_contracts.StemLeafResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	StemLeafConfig	required
valid_count	int	required
excluded_count	int	required
missing_count	int	required
leaf_unit	float	required
unit_chosen	bool	required
lines_per_stem	int	required
lines	tuple[StemLine,...]	required
omitted_lines	int	required
omitted_values	int	required
median	float	required

4.1.192 StemLine

tabelz.analysis.exploration_graph_contracts.StemLine

Common strict, immutable model baseline for public contracts.

Field	Type	Default
depth	str	required
stem	str	required

Field	Type	Default
leaves	str	required
count	int	required
omitted_leaves	int	0

4.1.193 CurvePoint

tabelz.analysis.extended_power.CurvePoint

Common strict, immutable model baseline for public contracts.

Field	Type	Default
x	float	required
y	float	required

y
Ge(ge=0.0); Le(le=1.0)

4.1.194 ExtendedPowerConfig

tabelz.analysis.extended_power.ExtendedPowerConfig

Strict prospective planning request.

Field	Type	Default
family	Literal['raw_t','raw_proportion','raw_anova','z','normal_variance','poisson_rate','equivalence','correlation','regression','anova_contrast','repeated_measures','paired_proportion','one_proportion']	required
solve_for	Literal['power','sample_size','effect','curve']	'sample_size'
design	Literal['one_sample','paired','independent']	'one_sample'
alternative	Literal['two-sided','less','greater']	'two-sided'
alpha	float	0.05
target_power	float	0.8
sample_size	int	30
effect_size	float|None	None
mean_difference	float|None	None
standard_deviation	float|None	None
known_sigma	float|None	None
proportion	float|None	None
null_proportion	float|None	None

Field	Type	Default
group_means	tuple[float,...]	()
group_sizes	tuple[int,...]	()
groups	int	3
ratio	float	1.0
variance_ratio	float|None	None
null_variance	float|None	None
poisson_rate	float|None	None
null_rate	float|None	None
exposure	float|None	None
true_difference	float|None	None
lower_margin	float|None	None
upper_margin	float|None	None
curve_parameter	Literal['sample_size','effect']	'sample_size'
curve_points	int	41
min_n	int	2
max_n	int	1000000
min_effect	float	0.001
max_effect	float	10.0
seed	int|None	None
addition	PowerAdditionSpec|None	None

alpha
Gt(gt=0.0); Lt(lt=0.5)

target_power
Gt(gt=0.0); Lt(lt=1.0)

sample_size
Ge(ge=2); Le(le=1000000)

groups
Ge(ge=2); Le(le=100)

ratio
Gt(gt=0.0); Le(le=100.0)

curve_points
Ge(ge=2); Le(le=201)

min_n
Ge(ge=2); Le(le=1000000)

max_n
Ge(ge=2); Le(le=1000000)

min_effect
Gt(gt=0.0); Le(le=100.0)

max_effect
Gt(gt=0.0); Le(le=100.0)

4.1.195 ExtendedPowerResult

`tabelz.analysis.extended_power.ExtendedPowerResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['raw_t','raw_proportion','raw_anova','z','normal_variance','poisson_rate','equivalence','correlation','regression','anova_contrast','repeated_measures','paired_proportion','one_proportion']	required
solve_for	Literal['power','sample_size','effect','curve']	required
design	Literal['one_sample','paired','independent']	required
alternative	Literal['two-sided','less','greater']	required
alpha	float	required
target_power	float	required
achieved_power	float	required
value	float	required
effect_size	float|None	required
sample_size	int|None	required
group_two_sample_size	int|None	required
curve_parameter	Literal['sample_size','effect']	'sample_size'
curve	tuple[CurvePoint,...]	()
table_payload	dict[str,object]	{}
warnings	tuple[str,...]	()
method_identity	str	'extended_power.prospective'
implementation_version	str	'1.3.0'
request_fingerprint	str	required
details	PowerAdditionDetails|None	None

4.1.196 EffectSolution

`tabelz.analysis.extended_power_addition_contracts.EffectSolution`

One branch of an effect solve.

Field	Type	Default
side	Literal['above_null','below_null']	required

Field	Type	Default
status	Literal['solved','not_found','unattainable']	required
value	float|None	None
power	float|None	None
detail	str	''
method	Literal['monotone_branch','grid_bisection']|None	None
search_low	float|None	None
search_high	float|None	None
headline	bool	False

power
Ge(ge=0.0); Le(le=1.0)

4.1.197 PowerAdditionDetails

tabelz.analysis.extended_power_addition_contracts.PowerAdditionDetails

Reference distribution, df, noncentrality and search disclosures.

Field	Type	Default
effect_label	str	required
reference	str	required
sample_size_label	str	required
total_sample_size	int|None	None
numerator_df	float|None	None
denominator_df	float|None	None
noncentrality	float|None	None
critical_value	float|None	None
attainable_alpha	float|None	None
search	Literal['none','monotone','first_qualifying']	'none'
search_upper_bound	int|None	None
stable_sample_size	int|None	None
solutions	tuple[EffectSolution,...]	()
attained_size_basis	str|None	None
search_lower_bound	int|None	None
search_step	int|None	None
search_scope	str|None	None
stability	Literal['not_applicable','assessed','not_assessed']	'not_applicable'

attainable_alpha
Ge(ge=0.0); Le(le=1.0)

4.1.198 PowerAdditionSpec

`tabelz.analysis.extended_power_addition_contracts.PowerAdditionSpec`

Family-specific planning parameters for the F151 families.

Field	Type	Default
correlation	float|None	None
null_correlation	float	0.0
second_correlation	float|None	None
correlation_method	Literal['fisher_z','spearman_bonett_wright','spearman_fhp','pearson_exact']|None	None
controls	int|None	None
allocation_ratios	tuple[int,...]|None	None
regression_mode	Literal['r2_from_zero','r2_increase']	'r2_from_zero'
r2_full	float|None	None
r2_reduced	float	0.0
tested_df	int|None	None
full_rank	int|None	None
contrast	tuple[float,...]	()
cohen_f	float|None	None
measurements	int|None	None
repeated_correlation	float|None	None
sphericity_epsilon	float	1.0
between_groups	int	1
repeated_effect	Literal['within','within_between','between']	'within'
partial_eta_squared	float|None	None
mcnemar_method	Literal['exact','connor']|None	None
discordant_10	float|None	None
discordant_01	float|None	None
proportion_method	Literal['exact','normal','normal_corrected']	'exact'

controls
Ge(ge=0); Le(le=10000)

allocation_ratios
MinLen(min_length=2); MaxLen(max_length=100)

tested_df
Ge(ge=1); Le(le=10000)

full_rank
Ge(ge=2); Le(le=100000)

contrast
MaxLen(max_length=100)

measurements
Ge(ge=2); Le(le=1000)

between_groups
Ge(ge=1); Le(le=1000)

partial_eta_squared
Ge(ge=0); Lt(lt=1)

4.1.199 FaConfig

`tabelz.analysis.factor_analysis.FaConfig`

Frozen, strict exploratory factor analysis request.

Field	Type	Default
`columns`	`tuple[str,...]`	`required`
`n_factors`	`int`	`required`
`rotation`	`Literal['none','varimax']`	`'none'`
`max_rows`	`int`	`1000000`
`max_cells`	`int`	`20000000`
`max_scatter_points`	`int`	`5000`

n_factors
Ge(ge=1); Le(le=16)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

max_scatter_points
Ge(ge=1); Le(le=5000)

4.1.200 FaResult

`tabelz.analysis.factor_analysis.FaResult`

Frozen, strict exploratory factor analysis outcome for transport.

Field	Type	Default
`columns`	`tuple[str,...]`	`required`
`n_factors`	`int`	`required`
`rotation`	`Literal['none','varimax']`	`required`
`status`	`Literal['converged','nonconverged','heywood']`	`required`
`means`	`tuple[float,...]`	`required`
`scales`	`tuple[float,...]`	`required`

Field	Type	Default
loadings	tuple[tuple[float,...],...]	required
unrotated_loadings	tuple[tuple[float,...],...]	required
uniquenesses	tuple[float,...]	required
communalities	tuple[float,...]	required
correlation_eigenvalues	tuple[float,...]	required
factor_common_variance	tuple[float,...]	required
score_weights	tuple[tuple[float,...],...]	required
scores	tuple[tuple[float,...],...]	required
row_ids	tuple[str,...]	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
warnings	tuple[str,...]	()

n_factors
Ge(ge=1)

source_count
Ge(ge=0)

excluded_count
Ge(ge=0)

missing_count
Ge(ge=0)

4.1.201 VolatilityForecastResult

tabelz.analysis.garch.VolatilityForecastResult

Field	Type	Default
response_column	str	required
model_label	str	required
distribution	str	required
sample_count	int	required
horizon	int	required
converged	bool	required
estimation_scale	float	required
volatility_unit	str	required
parameters	tuple[dict[str,object],...]	required
conditional_volatility	tuple[float,...]	required
forecast_mean	tuple[float,...]	required

Field	Type	Default
forecast_volatility	tuple[float,...]	required
persistence	float|None	required
forecast_rows	tuple[dict[str,object],...]	required
standardized_residuals	tuple[dict[str,object],...]	required
diagnostics	dict[str,object]	required
specification	dict[str,object]	required
specification_fingerprint	str	required
data_fingerprint	str	required
reproducibility	dict[str,str]	required
warnings	tuple[str,...]	()

4.1.202 GeneralizedLinearConfig

tabelz.analysis.generalized_linear_models_contracts.GeneralizedLinearConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
trials	str	''
factors	tuple[str,...]	()
covariates	tuple[str,...]	()
interactions	tuple[str,...]	()
quadratics	tuple[str,...]	()
quadratic_basis	Literal['centered','raw']	'centered'
nested	tuple[str,...]	()
offset	str	''
family	Literal['gaussian','binomial','poisson','negative_binomial','gamma','inverse_gaussian','tweedie']	'gaussian'
link	Literal['canonical','identity','log','logit','probit','cloglog','loglog','power','inverse','inverse_squared']	'canonical'
link_power	float	1.0
tweedie_power	float	1.5
theta	Literal['estimate','fixed']	'estimate'
theta_value	float	1.0
scale	Literal['default','ml']	'default'
covariance	Literal['model','hc0']	'model'
confidence_level	float	0.95

Field	Type	Default
lr_tests	bool	False
comparisons	Literal['holm','max_z','max_z_control']	'holm'
control_levels	dict[str,str]	factory:dict
cell_means	tuple[str,...]	()
simple_comparisons	tuple[str,...]	()
simple_family_scope	Literal['combined','per_stratum_prespecified']	'combined'
reference_covariates	dict[str,float]	factory:dict
factor_weights	dict[str,dict[str,float]]	factory:dict
reference_offset	float|None	None

factors
MaxLen(max_length=12)
covariates
MaxLen(max_length=32)
interactions
MaxLen(max_length=16)
quadratics
MaxLen(max_length=32)
nested
MaxLen(max_length=12)
theta_value
Gt(gt=0.0)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
cell_means
MaxLen(max_length=16)
simple_comparisons
MaxLen(max_length=16)

4.1.203 GeneralizedLinearResult

`tabelz.analysis.generalized_linear_models_contracts.GeneralizedLinearResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	GeneralizedLinearConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
family_label	str	required

Field	Type	Default
link_label	str	required
iterations	int	required
coefficients	tuple[GlmCoefficient,...]	required
statistic_kind	Literal['z','t']	required
type3	tuple[GlmTermTest,...]	required
type3_kind	Literal['chi-square','F']	required
lr_tests	tuple[GlmTermTest,...]	()
means	tuple[GlmMean,...]	()
comparisons	tuple[GlmComparison,...]	()
fit	GlmFitStatistics	required
artifact	GlmScoringArtifact	required
notes	tuple[str,...]	()
warnings	tuple[str,...]	()

4.1.204 GlmCoefficient

tabelz.analysis.generalized_linear_models_contracts.GlmCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
estimate	float	required
standard_error	float	required
statistic	float|None	required
p_value	float|None	required
lower	float	required
upper	float	required
exp_estimate	float|None	None
exp_lower	float|None	None
exp_upper	float|None	None

4.1.205 GlmComparison

tabelz.analysis.generalized_linear_models_contracts.GlmComparison

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factor	str	required
first	str	required
second	str	required

Field	Type	Default
difference	float	required
standard_error	float	required
statistic	float	required
p_value	float	required
holm_p_value	float	required
lower	float	required
upper	float	required
ratio	float|None	None
ratio_lower	float|None	None
ratio_upper	float|None	None
max_z_p_value	float|None	None
max_z_p_half_width	float|None	None
simultaneous_lower	float|None	None
simultaneous_upper	float|None	None
max_z_critical	float|None	None
max_z_critical_bracket	tuple[float,float]|None	None
calibration	MaxTCalibration|None	None

4.1.206 GlmFitStatistics

tabelz.analysis.generalized_linear_models_contracts.GlmFitStatistics

Common strict, immutable model baseline for public contracts.

Field	Type	Default
deviance	float	required
null_deviance	float	required
df_residual	int	required
df_null	int	required
pearson_chi_square	float	required
scale	float	required
scale_method	str	required
ml_scale	float|None	None
log_likelihood	float|None	None
aic	float|None	None
bic	float|None	None
parameters_counted	int|None	None
theta	float|None	None
theta_se	float|None	None
theta_method	str	''

4.1.207 GlmMean

`tabelz.analysis.generalized_linear_models_contracts.GlmMean`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`factor`	`str`	`required`
`level`	`str`	`required`
`link_mean`	`float`	`required`
`link_se`	`float`	`required`
`link_lower`	`float`	`required`
`link_upper`	`float`	`required`
`mean`	`float`	`required`
`mean_se`	`float`	`required`
`mean_lower`	`float|None`	`required`
`mean_upper`	`float|None`	`required`
`interval_note`	`str`	`''`

4.1.208 GlmScoringArtifact

`tabelz.analysis.generalized_linear_models_contracts.GlmScoringArtifact`

Everything the recorded score columns need to rebuild fitted means and residuals.

Field	Type	Default
`family`	`Literal['gaussian','binomial','poisson','negative_binomial','gamma','inverse_gaussian','tweedie']`	`required`
`link`	`str`	`required`
`link_power`	`float`	`required`
`tweedie_power`	`float`	`required`
`theta`	`float|None`	`required`
`response`	`str`	`required`
`trials`	`str`	`required`
`offset`	`str`	`required`
`factors`	`tuple[str,...]`	`required`
`covariates`	`tuple[str,...]`	`required`
`interactions`	`tuple[tuple[str,str],...]`	`required`
`quadratics`	`tuple[str,...]`	`()`
`centers`	`tuple[tuple[str,float],...]`	`()`
`nested_levels`	`tuple[tuple[str,str,tuple[tuple[str,tuple[str,...]],...]],...]`	`()`
`levels`	`tuple[tuple[str,tuple[str,...]],...]`	`required`
`coefficients`	`tuple[float,...]`	`required`

Field	Type	Default
covariance	tuple[tuple[float,...],...]	required
critical	float	required
scale	float	required
prediction_intervals	bool	required

4.1.209 GlmTermTest

tabelz.analysis.generalized_linear_models_contracts.GlmTermTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
df	int	required
statistic	float	required
denominator_df	int|None	None
p_value	float	required

4.1.210 AnalysisGolden

tabelz.analysis.goldens.AnalysisGolden

Stable fingerprint projections plus all resolved semantic payloads.

Field	Type	Default
envelope_projection	EnvelopeFingerprint	required
envelope_fingerprint	str	required
component_projections	tuple[ScalarComponentFingerprint|TableComponentFingerprint|IntervalComponentFingerprint|WarningComponentFingerprint|DiagnosticComponentFingerprint|ScatterPlotComponentFingerprint|HistogramPlotComponentFingerprint,...]	()
receipt_projection	ProvenanceReceipt	required
receipt_fingerprint	str	required
environment	EngineProvenance	required
payloads	tuple[TablePayloadProjection|ScatterPayloadProjection|HistogramPayloadProjection,...]	()

4.1.211 GuidedScoreResult

tabelz.analysis.guided_scoring.GuidedScoreResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
artifact	GuidedScoringArtifact	required
row_ids	tuple[str,...]	required
predictions	tuple[float,...]	required
lower	tuple[float|None,...]	required
upper	tuple[float|None,...]	required
unavailable_row_ids	tuple[str,...]	required
excluded_row_ids	tuple[str,...]	required
note	str	required

4.1.212 GuidedScoringArtifact

tabelz.analysis.guided_scoring.GuidedScoringArtifact

Common strict, immutable model baseline for public contracts.

Field	Type	Default
version	Literal['1.0.0']	'1.0.0'
identity	Literal['estimation','locked_evaluation']	required
response	str	required
family	Literal['ols_hc3','logistic','poisson','negative_binomial','ols','ridge','lasso']	required
features	tuple[ScoringFeature,...]	required
coefficients	tuple[float,...]	required
covariance	tuple[tuple[float,...],...]	()
dispersion	float|None	None
reference_df	float|None	None
exposure	str|None	None
event	str|None	None
confidence_level	float	0.95
imputation	Literal['drop','mean']	'drop'
fitted_row_ids	tuple[str,...]	required
evaluation_row_ids	tuple[str,...]	()
provenance	dict[str,Any]	factory:dict

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.213 ScoringFeature

tabelz.analysis.guided_scoring.ScoringFeature

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
level	str|None	None
levels	tuple[str,...]	()
fill	float|None	None
center	float	0.0
scale	float	1.0

scale
Gt(gt=0.0)

4.1.214 GuidedScoringConfig

tabelz.analysis.guided_scoring_columns.GuidedScoringConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
fit_key	str	required
artifact_json	str	required

4.1.215 HierarchicalClusteringConfig

tabelz.analysis.hierarchical_clustering.HierarchicalClusteringConfig

Frozen, strict clustering request: mode, columns, linkage, and limits.

Field	Type	Default
columns	tuple[str,...]	required
mode	Literal['rows','variables']	'rows'
linkage	Literal['ward','average','complete']	'ward'
standardize	bool	True
cut_k	int|None	None
max_rows	int	2000
max_pairwise_cells	int	12500000

cut_k
Ge(ge=2)
max_rows
Ge(ge=2); Le(le=5000)
max_pairwise_cells
Ge(ge=1); Le(le=12500000)

4.1.216 HierarchicalClusteringResult

`tabelz.analysis.hierarchical_clustering.HierarchicalClusteringResult`

Frozen, strict clustering outcome for transport and rendering.

Field	Type	Default
columns	tuple[str,...]	required
mode	Literal['rows','variables']	required
linkage	Literal['ward','average','complete']	required
distance	Literal['euclidean','correlation']	required
standardize	bool	required
cut_k	int|None	required
means	tuple[float,...]	required
scales	tuple[float,...]	required
merge_joins	tuple[tuple[int,int],...]	required
merge_heights	tuple[float,...]	required
merge_sizes	tuple[int,...]	required
leaf_order	tuple[int,...]	required
cluster_labels	tuple[int,...]	required
cluster_counts	tuple[int,...]	required
item_ids	tuple[str,...]	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)

excluded_count
Ge(ge=0)

missing_count
Ge(ge=0)

4.1.217 HypothesisResult

`tabelz.analysis.hypothesis_contracts.HypothesisResult`

Field	Type	Default
payload	dict[str,object]	required

4.1.218 HypothesisTestConfig

`tabelz.analysis.hypothesis_contracts.HypothesisTestConfig`

Field	Type	Default
test	str	required
response	str	''
group	str	''
second_response	str	''
second_group	str	''
target	float	0.0
success_value	str	''
alpha	float	0.05
alternative	Literal['two-sided','less','greater']	'two-sided'
method	str	''
posthoc	bool	True
expected_proportions	tuple[float,...]	()
stratum	str	''
category_order	tuple[str,...]	()
scores	tuple[float,...]	()
seed	int	0
permutation_draws	int	0
columns	tuple[str,...]	()
distribution	str	''
parameters	tuple[float,...]	()
cut_point	float|None	None
trim	int|None	None
control	str	''
followup	str	''

4.1.219 PowerAnalysisConfig

tabelz.analysis.hypothesis_contracts.PowerAnalysisConfig

Field	Type	Default
analysis	str	required
solve_for	Literal['sample_size','power']	'sample_size'
effect_size	float	0.5
alpha	float	0.05
target_power	float	0.8
sample_size	int	30
groups	int	3
ratio	float	1.0
alternative	Literal['two-sided','less','greater']	'two-sided'

4.1.220 ItemAnalysisConfig

`tabelz.analysis.item_analysis.ItemAnalysisConfig`

Frozen, strict item-analysis request.

Field	Type	Default
`items`	`tuple[str,...]`	`required`
`reverse_items`	`tuple[str,...]`	`()`
`scale_low`	`float|None`	`None`
`scale_high`	`float|None`	`None`
`max_rows`	`int`	`1000000`
`max_cells`	`int`	`20000000`

max_rows
Ge(ge=1); Le(le=1000000)
max_cells
Ge(ge=1); Le(le=20000000)

4.1.221 ItemAnalysisResult

`tabelz.analysis.item_analysis.ItemAnalysisResult`

Frozen, strict item-analysis outcome for transport and rendering.

Field	Type	Default
`items`	`tuple[str,...]`	`required`
`reverse_items`	`tuple[str,...]`	`required`
`scale_low`	`float|None`	`required`
`scale_high`	`float|None`	`required`
`cronbach_alpha`	`float`	`required`
`standardized_alpha`	`float|None`	`required`
`item_summaries`	`tuple[ItemSummary,...]`	`required`
`correlations`	`tuple[tuple[float|None,...],...]`	`required`
`row_ids`	`tuple[str,...]`	`required`
`source_count`	`int`	`required`
`excluded_count`	`int`	`required`
`missing_count`	`int`	`required`
`warnings`	`tuple[str,...]`	`()`

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)

4.1.222 ItemSummary

`tabelz.analysis.item_analysis.ItemSummary`

Per-item descriptive and diagnostic summary.

Field	Type	Default
item	str	required
reverse_coded	bool	required
adjusted_mean	float	required
adjusted_sd	float	required
corrected_item_total	float|None	required
alpha_if_deleted	float|None	required

4.1.223 KaplanMeierRow

`tabelz.analysis.life_distribution_analysis.KaplanMeierRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
time	float	required
at_risk	float	required
failures	float	required
censored	float	required
survival	float	required
lower	float|None	None
upper	float|None	None

4.1.224 LifeCounts

`tabelz.analysis.life_distribution_analysis.LifeCounts`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
source_count	int	required
excluded_count	int	required
missing_count	int	required
zero_frequency_rows	int	required
rows_used	int	required
exact	float	required
right	float	required
left	float	required

Field	Type	Default
interval	float	required

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
zero_frequency_rows
Ge(ge=0)
rows_used
Ge(ge=0)
exact
Ge(ge=0.0)
right
Ge(ge=0.0)
left
Ge(ge=0.0)
interval
Ge(ge=0.0)

4.1.225 LifeDistributionConfig

tabelz.analysis.life_distribution_analysis.LifeDistributionConfig

Where the censored life records are and how they are coded.

Field	Type	Default
layout	Literal['arbitrary','right_censored']	'arbitrary'
start	str	required
end	str|None	None
status	str|None	None
frequency	str|None	None
family	Literal['exponential','weibull','log normal','loglogistic','normal','logistic','sev']	'weibull'
threshold	float	0.0
confidence_level	float	0.95
percents	tuple[float,...]	(1.0,5.0,10.0,50.0)

start
MinLen(min_length=1)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.226 LifeDistributionResult

`tabelz.analysis.life_distribution_analysis.LifeDistributionResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['exponential','weibull','lognormal','loglogistic','normal','logistic','sev']	required
threshold	float	required
confidence_level	float	required
counts	LifeCounts	required
uninformative_weight	float	required
parameters	tuple[LifeParameter,...]	required
log_likelihood	float	required
aic	float	required
bic	float	required
covariance	tuple[tuple[float,...],...]	required
percentiles	tuple[LifePercentile,...]	required
nonparametric	NonparametricLife	required
curve	tuple[tuple[float,float],...]	required
warnings	tuple[str,...]	()

uninformative_weight
Ge(ge=0.0)

4.1.227 LifeParameter

`tabelz.analysis.life_distribution_analysis.LifeParameter`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
standard_error	float|None	None
lower	float|None	None
upper	float|None	None
note	str	''

4.1.228 LifePercentile

`tabelz.analysis.life_distribution_analysis.LifePercentile`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
percent	float	required
estimate	float	required
standard_error	float|None	None
lower	float|None	None
upper	float|None	None

4.1.229 NonparametricLife

tabelz.analysis.life_distribution_analysis.NonparametricLife

Kaplan-Meier or Turnbull; Turnbull carries no bands (marked unavailable).

Field	Type	Default
method	Literal['kaplan_meier','turnbull']	required
kaplan_meier	tuple[KaplanMeierRow,...]	()
turnbull	tuple[TurnbullRow,...]	()
bands_available	bool	required
iterations	int	0
max_gradient	float|None	None
monotone	bool	True
unidentified_tail	bool	False

4.1.230 NonparametricLifeConfig

tabelz.analysis.life_distribution_analysis.NonparametricLifeConfig

Where the censored life records are and how they are coded.

Field	Type	Default
layout	Literal['arbitrary','right_censored']	'arbitrary'
start	str	required
end	str|None	None
status	str|None	None
frequency	str|None	None
confidence_level	float	0.95

start
MinLen(min_length=1)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.231 NonparametricLifeResult

tabelz.analysis.life_distribution_analysis.NonparametricLifeResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
confidence_level	float	required
counts	LifeCounts	required
estimate	NonparametricLife	required
warnings	tuple[str, ...]	()

4.1.232 TurnbullRow

tabelz.analysis.life_distribution_analysis.TurnbullRow

One innermost interval; “None“ marks an unbounded end.

Field	Type	Default
low	float|None	required
high	float|None	required
low_closed	bool	required
mass	float	required
survival_before	float	required
survival_after	float	required

mass
Ge(ge=0.0)

4.1.233 LifeDistributionIdConfig

tabelz.analysis.life_distribution_id.LifeDistributionIdConfig

Where the censored life records are and how they are coded.

Field	Type	Default
layout	Literal['arbitrary','right_censored']	'arbitrary'
start	str	required
end	str|None	None
status	str|None	None
frequency	str|None	None
families	tuple[Literal['exponential','weibull','lognormal','loglogistic','normal','logistic','sev'],...]	('exponential','weibull','lognormal','loglogistic','normal','logistic','sev')
threshold	float	0.0
confidence_level	float	0.95

start
MinLen(min_length=1)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.234 LifeDistributionIdResult

`tabelz.analysis.life_distribution_id.LifeDistributionIdResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`threshold`	`float`	`required`
`confidence_level`	`float`	`required`
`counts`	`LifeCounts`	`required`
`plotting_positions`	`str`	`required`
`plotted_points`	`int`	`required`
`thinned`	`bool`	`required`
`ad_measure`	`Literal['anderson_darling','restricted_domain_ad']`	`'anderson_darling'`
`ad_domain`	`str`	`'thewholesupport'`
`families`	`tuple[ProbabilityPlotFamily,...]`	`required`
`nonparametric`	`NonparametricLife`	`required`
`warnings`	`tuple[str,...]`	`()`

plotted_points
Ge(ge=0)

4.1.235 ProbabilityPlotFamily

`tabelz.analysis.life_distribution_id.ProbabilityPlotFamily`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`family`	`Literal['exponential','weibull','lognormal','loglogistic','normal','logistic','sev']`	`required`
`fitted`	`bool`	`required`
`message`	`str`	`''`
`parameters`	`tuple[LifeParameter,...]`	`()`
`log_likelihood`	`float|None`	`None`
`anderson_darling`	`float|None`	`None`
`ad_domain_probability`	`float|None`	`None`
`ad_note`	`str`	`''`
`correlation`	`float|None`	`None`
`percentiles`	`tuple[LifePercentile,...]`	`()`

Field	Type	Default
points	tuple[tuple[float,float],...]	()
line	tuple[tuple[float,float],...]	()
x_label	str	''

4.1.236 LifeCoefficient

tabelz.analysis.life_regression.LifeCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
estimate	float	required
standard_error	float	required
z_value	float	required
lower	float	required
upper	float	required

4.1.237 LifePrediction

tabelz.analysis.life_regression.LifePrediction

Common strict, immutable model baseline for public contracts.

Field	Type	Default
setting	str	required
percent	float	required
estimate	float	required
standard_error	float|None	None
lower	float|None	None
upper	float|None	None
extrapolated	bool	False

4.1.238 LifeRegressionConfig

tabelz.analysis.life_regression.LifeRegressionConfig

Where the censored life records are and how they are coded.

Field	Type	Default
layout	Literal['arbitrary','right_censored']	'arbitrary'
start	str	required
end	str|None	None

Field	Type	Default
status	str|None	None
frequency	str|None	None
family	Literal['exponential','weibull','log normal','loglogistic','normal','logis tic','sev']	'weibull'
threshold	float	0.0
numeric_predictors	tuple[str,...]	()
categorical_predic tors	tuple[str,...]	()
transform	Literal['none','arrhenius','inverse_p ower']	'none'
transform_column	str|None	None
temperature_unit	Literal['celsius','kelvin']	'celsius'
prediction_rows	tuple[str,...]	()
percents	tuple[float,...]	(10.0,50.0)
confidence_level	float	0.95

start
MinLen(min_length=1)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.239 LifeRegressionResult

tabelz.analysis.life_regression.LifeRegressionResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['exponential','weibull','log normal','loglogistic','normal','logis tic','sev']	required
threshold	float	required
confidence_level	float	required
transform	Literal['none','arrhenius','inverse_p ower']	required
transform_column	str|None	required
transform_formula	str	required
counts	LifeCounts	required
coefficients	tuple[LifeCoefficient,...]	required
dispersion	LifeCoefficient|None	required
shape	LifeCoefficient|None	required
log_likelihood	float	required

Field	Type	Default
factor_levels	tuple[tuple[str,tuple[str,...]],...]	required
covariance	tuple[tuple[float,...],...]	required
predictions	tuple[LifePrediction,...]	required
warnings	tuple[str,...]	()

4.1.240 GroupedComparison

tabelz.analysis.life_table_comparison.GroupedComparison

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method	str	required
statistic	float|None	required
df	int	required
p_value	float|None	required
strata	tuple[StratumScore,...]	required
withheld	str	''

4.1.241 LifeTableMedian

tabelz.analysis.life_table_comparison.LifeTableMedian

Common strict, immutable model baseline for public contracts.

Field	Type	Default
status	Literal['estimated','beyond_last_boundary','not_reached']	required
value	float|None	None
bound	float|None	None
statement	str	required

4.1.242 StratumScore

tabelz.analysis.life_table_comparison.StratumScore

Common strict, immutable model baseline for public contracts.

Field	Type	Default
stratum	str|None	required
subjects	int	required
score_sum	float	required
mean_score	float	required

4.1.243 LifeTableConfig

`tabelz.analysis.life_tables.LifeTableConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`time`	`str`	`required`
`status`	`str`	`required`
`event_value`	`str`	`'1'`
`censored_value`	`str`	`'0'`
`interval_width`	`float|None`	`None`
`last_boundary`	`float|None`	`None`
`boundaries`	`tuple[float,...]`	`()`
`strata`	`str|None`	`None`

time
MinLen(min_length=1)

status
MinLen(min_length=1)

event_value
MinLen(min_length=1)

censored_value
MinLen(min_length=1)

interval_width
Gt(gt=0.0)

last_boundary
Gt(gt=0.0)

4.1.244 LifeTableResult

`tabelz.analysis.life_tables.LifeTableResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`LifeTableConfig`	`required`
`edges`	`tuple[float,...]`	`required`
`strata`	`tuple[LifeTableStratum,...]`	`required`
`source_count`	`int`	`required`
`excluded_count`	`int`	`required`
`missing_count`	`int`	`required`
`analyzed_count`	`int`	`required`
`notes`	`tuple[str,...]`	`()`
`warnings`	`tuple[str,...]`	`()`

Field	Type	Default
comparison	GroupedComparison|None	None

4.1.245 LifeTableRow

tabelz.analysis.life_tables.LifeTableRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
start	float	required
end	float|None	required
width	float|None	required
entering	int	required
withdrawn	int	required
events	int	required
at_risk	float	required
q	float	required
p	float	required
survival_start	float	required
survival_start_se	float|None	required
survival_end	float|None	required
survival_end_se	float|None	required
density	float|None	required
density_se	float|None	required
hazard	float|None	required
hazard_se	float|None	required
unavailable	tuple[str,...]	()

4.1.246 LifeTableStratum

tabelz.analysis.life_tables.LifeTableStratum

Common strict, immutable model baseline for public contracts.

Field	Type	Default
stratum	str|None	required
subjects	int	required
events	int	required
withdrawn	int	required
rows	tuple[LifeTableRow,...]	required
median	LifeTableMedian|None	None

4.1.247 LinearEffectsConfig

`tabelz.analysis.linear_effects.LinearEffectsConfig`

Immutable additive OLS request.

Field	Type	Default
`response_column`	`str`	`required`
`factor_columns`	`tuple[str,...]`	`()`
`covariate_columns`	`tuple[str,...]`	`()`
`comparisons`	`Literal['holm','tukey','dunnett']`	`'holm'`
`control_levels`	`dict[str,str]`	`factory:dict`
`confidence_level`	`float`	`0.95`
`max_rows`	`int`	`1000000`

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.248 LinearEffectsContrast

`tabelz.analysis.linear_effects.LinearEffectsContrast`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`factor`	`str`	`required`
`left_level`	`str`	`required`
`right_level`	`str`	`required`
`difference`	`float`	`required`
`standard_error`	`float`	`required`
`t_statistic`	`float`	`required`
`p_value`	`float`	`required`
`holm_p_value`	`float`	`required`
`lower`	`float`	`required`
`upper`	`float`	`required`
`family_p_value`	`float|None`	`None`
`family_lower`	`float|None`	`None`
`family_upper`	`float|None`	`None`
`family_p_half_width`	`float|None`	`None`
`family_unresolved`	`bool`	`False`

4.1.249 LinearEffectsMean

`tabelz.analysis.linear_effects.LinearEffectsMean`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factor	str	required
level	str	required
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required
conditioning_values	dict[str,float]	required

4.1.250 LinearEffectsResult

tabelz.analysis.linear_effects.LinearEffectsResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_column	str	required
observations	int	required
source_count	int	required
missing_count	int	required
excluded_count	int	required
design_rank	int	required
residual_df	int	required
residual_variance	float	required
coefficients	dict[str,float]	required
terms	tuple[LinearEffectsTerm,...]	required
term_tests	tuple[LinearEffectsTermTest,...]	required
adjusted_means	tuple[LinearEffectsMean,...]	required
contrasts	tuple[LinearEffectsContrast,...]	required
comparison_families	tuple[ComparisonFamilyRecord,...]	()
covariate_means	dict[str,float]	required
warnings	tuple[str,...]	()
method	str	'linear_effects.ols_type_ii_adjusted_means'
retained_row_ids	tuple[str,...]	()
population_fingerprint	str|None	None

4.1.251 LinearEffectsTerm

tabelz.analysis.linear_effects.LinearEffectsTerm

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
kind	str	required
columns	tuple[str,...]	required
reference_level	str|None	None
levels	tuple[str,...]	()

4.1.252 LinearEffectsTermTest

tabelz.analysis.linear_effects.LinearEffectsTermTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
numerator_df	int	required
denominator_df	int	required
sum_sq	float	required
mean_sq	float	required
f_statistic	float	required
p_value	float	required

4.1.253 GeneralLinearContrast

tabelz.analysis.linear_model_terms.GeneralLinearContrast

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factor	str	required
left_level	str	required
right_level	str	required
difference	float	required
standard_error	float	required
t_statistic	float	required
p_value	float	required
holm_p_value	float	required
lower	float	required
upper	float	required
family_p_value	float|None	None
family_lower	float|None	None
family_upper	float|None	None

Field	Type	Default
family_p_half_width	float|None	None
family_unresolved	bool	False

4.1.254 GeneralLinearMean

tabelz.analysis.linear_model_terms.GeneralLinearMean

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factor	str	required
level	str	required
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required
conditioning_values	dict[str,float]	required
grid_weighting	str	required

4.1.255 GeneralLinearResult

tabelz.analysis.linear_model_terms.GeneralLinearResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_column	str	required
observations	int	required
source_count	int	required
missing_count	int	required
excluded_count	int	required
design_rank	int	required
residual_df	int	required
residual_variance	float	required
weighted	bool	required
weight_column	str|None	required
coefficients	dict[str,float]	required
terms	tuple[GeneralLinearTerm,...]	required
term_tests	tuple[GeneralLinearTermTest,...]	required
adjusted_means	tuple[GeneralLinearMean,...]	required
contrasts	tuple[GeneralLinearContrast,...]	required
comparison_families	tuple[ComparisonFamilyRecord,...]	()

Field	Type	Default
covariate_means	dict[str,float]	required
warnings	tuple[str,...]	()
method	str	'general_linear.ols_wls_type_ii_iii_sum'
retained_row_ids	tuple[str,...]	()
population_fingerprint	str|None	None

4.1.256 GeneralLinearTerm

tabelz.analysis.linear_model_terms.GeneralLinearTerm

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
kind	str	required
columns	tuple[str,...]	required
levels	tuple[str,...]	()
reference_level	str|None	None
degrees_of_freedom	int	required

4.1.257 GeneralLinearTermTest

tabelz.analysis.linear_model_terms.GeneralLinearTermTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
numerator_df	int	required
denominator_df	int	required
sum_sq	float	required
mean_sq	float	required
f_statistic	float	required
p_value	float	required
convention	str	required

4.1.258 GeneralLinearConfig

tabelz.analysis.linear_model_terms_contracts.GeneralLinearConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_column	str	required
factor_columns	tuple[str,...]	()
covariate_columns	tuple[str,...]	()
interactions	tuple[tuple[str,str],...]	()
quadratics	tuple[str,...]	()
nested	tuple[tuple[str,str],...]	()
terms	tuple[GeneralLinearTermSpec,...]	()
weight_column	str|None	None
weight_kind	Literal['precision']	'precision'
test_type	Literal['II','III']	'III'
comparisons	Literal['holm','tukey','dunnett']	'holm'
control_levels	dict[str,str]	factory:dict
simple_effects	tuple[tuple[str,str],...]	()
simple_family_scope	Literal['combined','per_stratum_presp ecified']	'combined'
combined_factors	tuple[str,...]	()
confidence_level	float	0.95
max_rows	int	1000000

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.259 GeneralLinearTermSpec

`tabelz.analysis.linear_model_terms_contracts.GeneralLinearTermSpec`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['main','interaction','quadrat ic','nested']	required
columns	tuple[str,...]	required

4.1.260 LogitParameter

`tabelz.analysis.loglinear_contracts.LogitParameter`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
logit	str	required
term	str	required

Field	Type	Default
label	str	required
source	str	required
estimate	float|None	required
standard_error	float|None	required

4.1.261 LoglinearCell

tabelz.analysis.loglinear_contracts.LoglinearCell

Common strict, immutable model baseline for public contracts.

Field	Type	Default
levels	tuple[str,...]	required
observed	float	required
expected	float|None	required
structural	bool	required
residual	float|None	required
adjusted_residual	float|None	required
deviance_residual	float|None	required

4.1.262 LoglinearConfig

tabelz.analysis.loglinear_contracts.LoglinearConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factors	tuple[str,...]	required
count	str	''
terms	tuple[str,...]	()
sampling	Literal['poisson','multinomial','product_multinomial']	'poisson'
fixed_margins	tuple[str,...]	()
dependent	tuple[str,...]	()
cell_structure	str	''
structural_zeros	tuple[str,...]	()
pseudocount	bool	False
coding	Literal['last','first']	'last'
confidence_level	float	0.95
reference_levels	tuple[str,...]	()
intervals	Literal['wald','wald_and_profile']	'wald'
functionals	tuple[str,...]	()

factors
MinLen(min_length=2); MaxLen(max_length=6)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.263 LoglinearFit

`tabelz.analysis.loglinear_contracts.LoglinearFit`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
generators	tuple[str,...]	required
terms	tuple[str,...]	required
parameters	tuple[LoglinearParameter,...]	required
cells	tuple[LoglinearCell,...]	required
allowable_cells	int	required
structural_zero_cells	int	required
rank	int	required
df	int	required
deviance	float	required
pearson	float	required
deviance_p	float|None	required
pearson_p	float|None	required
log_likelihood	float	required
boundary	bool	required
boundary_cells	int	required
aliased	tuple[str,...]	()

4.1.264 LoglinearFunctional

`tabelz.analysis.loglinear_contracts.LoglinearFunctional`

An estimable functional c'beta of the coefficients (#147).

Field	Type	Default
name	str	required
weights	tuple[tuple[str,float],...]	required
estimable	bool	required
estimate	float|None	None
standard_error	float|None	None
z_value	float|None	None
p_value	float|None	None
lower	float|None	None

Field	Type	Default
upper	float|None	None
profile_lower	float|None	None
profile_upper	float|None	None
status	str	''
profile_status	str	''

4.1.265 LoglinearParameter

tabelz.analysis.loglinear_contracts.LoglinearParameter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
label	str	required
estimate	float|None	required
standard_error	float|None	required
z_value	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required
status	str	''
profile_lower	float|None	None
profile_upper	float|None	None
profile_status	str	''

4.1.266 LoglinearResult

tabelz.analysis.loglinear_contracts.LoglinearResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	LoglinearConfig	required
factor_levels	tuple[tuple[str,tuple[str,...]],...]	required
reference_levels	tuple[tuple[str,str],...]	required
total	float	required
fit	LoglinearFit	required
logit	tuple[LogitParameter,...]	()
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
functionals	tuple[LoglinearFunctional,...]	()

4.1.267 LoglinearSelectionConfig

`tabelz.analysis.loglinear_contracts.LoglinearSelectionConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`factors`	`tuple[str,...]`	`required`
`count`	`str`	`''`
`terms`	`tuple[str,...]`	`()`
`sampling`	`Literal['poisson','multinomial','product_multinomial']`	`'poisson'`
`fixed_margins`	`tuple[str,...]`	`()`
`dependent`	`tuple[str,...]`	`()`
`cell_structure`	`str`	`''`
`structural_zeros`	`tuple[str,...]`	`()`
`pseudocount`	`bool`	`False`
`coding`	`Literal['last','first']`	`'last'`
`confidence_level`	`float`	`0.95`
`reference_levels`	`tuple[str,...]`	`()`
`intervals`	`Literal['wald','wald_and_profile']`	`'wald'`
`functionals`	`tuple[str,...]`	`()`
`alpha`	`float`	`0.05`
`highest_order_only`	`bool`	`False`

factors
MinLen(min_length=2); MaxLen(max_length=6)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
alpha
Gt(gt=0.0); Lt(lt=1.0)

4.1.268 LoglinearSelectionResult

`tabelz.analysis.loglinear_contracts.LoglinearSelectionResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`LoglinearSelectionConfig`	`required`
`factor_levels`	`tuple[tuple[str,tuple[str,...]],...]`	`required`
`steps`	`tuple[SelectionStep,...]`	`required`
`final`	`LoglinearFit`	`required`
`notes`	`tuple[str,...]`	`()`
`warnings`	`tuple[str,...]`	`()`

4.1.269 SelectionCandidate

`tabelz.analysis.loglinear_contracts.SelectionCandidate`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`term`	`str`	`required`
`order`	`int`	`required`
`lr_chi_square`	`float`	`required`
`df`	`int`	`required`
`p_value`	`float|None`	`required`
`status`	`str`	`''`

4.1.270 SelectionStep

`tabelz.analysis.loglinear_contracts.SelectionStep`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`step`	`int`	`required`
`model`	`tuple[str,...]`	`required`
`candidates`	`tuple[SelectionCandidate,...]`	`required`
`removed`	`str|None`	`required`

4.1.271 ManovaAdjustedMean

`tabelz.analysis.manova.ManovaAdjustedMean`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`response`	`str`	`required`
`factor`	`str`	`required`
`level`	`str`	`required`
`estimate`	`float`	`required`
`standard_error`	`float`	`required`
`lower`	`float`	`required`
`upper`	`float`	`required`
`conditioning_values`	`dict[str,float]`	`required`
`grid_weighting`	`str`	`required`

4.1.272 ManovaJointTest

`tabelz.analysis.manova.ManovaJointTest`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
statistic	str	required
value	float	required
f_statistic	float	required
numerator_df	float	required
denominator_df	float	required
p_value	float	required
convention	str	required

4.1.273 ManovaResponseTermTest

tabelz.analysis.manova.ManovaResponseTermTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
term	str	required
numerator_df	int	required
denominator_df	int	required
sum_sq	float	required
f_statistic	float	required
p_value	float	required
holm_p_value	float	required
convention	str	required

4.1.274 ManovaResult

tabelz.analysis.manova.ManovaResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_columns	tuple[str,...]	required
observations	int	required
source_count	int	required
missing_count	int	required
excluded_count	int	required
design_rank	int	required
residual_df	int	required

Field	Type	Default
residual_covariance_df	int	required
residual_covariance_rank	int	required
residual_covariance	tuple[tuple[float,...],...]	required
group_counts	dict[str,dict[str,int]]	required
coefficients	dict[str,dict[str,float]]	required
terms	tuple[dict[str,object],...]	required
joint_tests	tuple[ManovaJointTest,...]	required
response_tests	tuple[ManovaResponseTermTest,...]	required
adjusted_means	tuple[ManovaAdjustedMean,...]	()
comparisons	tuple[ManovaComparison,...]	()
comparison_families	tuple[ManovaComparisonFamily,...]	()
plots	dict[str,object]	factory:dict
warnings	tuple[str,...]	()
method	str	'manova.ols_pillai_wilks_sum'
retained_row_ids	tuple[str,...]	()
population_fingerprint	str|None	None

4.1.275 ManovaConfig

tabelz.analysis.manova_contracts.ManovaConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_columns	tuple[str,...]	required
factor_columns	tuple[str,...]	()
covariate_columns	tuple[str,...]	()
terms	tuple[GeneralLinearTermSpec,...]	()
test_type	Literal['II','III']	'III'
confidence_level	float	0.95
max_rows	int	1000000
comparisons	Literal['none','tukey','dunnett']	'none'
control_levels	dict[str,str]	factory:dict
allocate_across_families	bool	False

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.276 ManovaComparison

`tabelz.analysis.manova_followups.ManovaComparison`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
factor	str	required
first	str	required
second	str	required
difference	float	required
standard_error	float	required
statistic	float	required
residual_df	int	required
family_p_value	float	required
family_lower	float	required
family_upper	float	required
numerical_p_half_width	float	required
allocated_p_value	float|None	None
allocated_p_half_width	float|None	None

4.1.277 ManovaComparisonFamily

`tabelz.analysis.manova_followups.ManovaComparisonFamily`

What one factor's max-t family compared and how it was calibrated.

Field	Type	Default
factor	str	required
procedure	Literal['tukey','dunnett']	required
control_level	str|None	required
levels	tuple[str,...]	required
averaged_over	tuple[str,...]	required
covariate_values	dict[str,float]	required
residual_df	int	required
critical_value	float|None	required
critical_bracket	tuple[float,float]|None	required
calibration	FamilyCalibration|None	required
notes	tuple[str,...]	()

Field	Type	Default
alpha	float	0.05
response	str	required
allocation_count	int	required
allocation_note	str	required

alpha
Gt(gt=0); Lt(lt=0.5)

4.1.278 AcceptancePlan

tabelz.analysis.manufacturing.AcceptancePlan

One single-sample attributes plan: inspect “n“, accept through “c“.

Field	Type	Default
sample_size	int	required
accept_number	int	required
aql	float	required
ltpd	float	required
producer_risk	float	required
consumer_risk	float	required
probability_accept_aql	float	required
probability_accept_ltpd	float	required
lot_size	int|None	None

4.1.279 ReliabilityResult

tabelz.analysis.manufacturing.ReliabilityResult

Field	Type	Default
distribution	str	required
n	int	required
failures	int	required
censored	int	required
shape	float	required
scale	float	required
confidence_level	float	required
shape_lower	float|None	required
shape_upper	float|None	required
scale_lower	float|None	required
scale_upper	float|None	required

Field	Type	Default
log_likelihood	float	required
b10_life	float	required
b10_lower	float|None	required
b10_upper	float|None	required
median_life	float	required
probability_points	tuple[dict[str,float|bool],...]	required
warning	str|None	required

4.1.280 ToleranceInterval

tabelz.analysis.manufacturing.ToleranceInterval

Field	Type	Default
distribution	str	required
sided	Literal['one_lower','one_upper','two_sided']	required
content	float	required
confidence	float	required
n	int	required
lower	float|None	required
upper	float|None	required
factor	float|None	required
method	str	required
achieved_confidence	float|None	None

4.1.281 MaxTCalibration

tabelz.analysis.max_t_calibration.MaxTCalibration

How a max-t critical value and its p-values were calibrated, and how well.

Field	Type	Default
method	str	'spherical-radialquasi-MonteCarlooverthedirection,exactradialFtail'
replications	int	required
points_per_replication	int	required
total_points	int	required
seeds	tuple[int,...]	required
target_half_width	float	required

Field	Type	Default
p_half_width	float	required
critical_probability_half_width	float	required
status	Literal['converged','budgetexhausted']	required

replications
Ge(ge=2)
points_per_replication
Ge(ge=1)
total_points
Ge(ge=1)
p_half_width
Ge(ge=0.0)
critical_probability_half_width
Ge(ge=0.0)

4.1.282 MetaAnalysisConfig

tabelz.analysis.meta_analysis_contracts.MetaAnalysisConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
data	Literal['generic','continuous','binary']	'generic'
study	str|None	None
effect	str|None	None
standard_error	str|None	None
n1	str|None	None
mean1	str|None	None
sd1	str|None	None
n2	str|None	None
mean2	str|None	None
sd2	str|None	None
events1	str|None	None
total1	str|None	None
events2	str|None	None
total2	str|None	None
measure	Literal['md','hedges_g','cohens_d','log_or','log_rr','rd']	'hedges_g'
correction	float	0.5
model	Literal['random','fixed']	'random'

Field	Type	Default
tau2	Literal['reml','dl','pm','sj']	'reml'
inference	Literal['modified_hk','hk','normal']	'modified_hk'
confidence_level	float	0.95
subgroup	str|None	None
moderators	tuple[str,...]	()
moderator_factors	tuple[str,...]	()
order	str|None	None
trimfill	Literal['l0','r0']	'l0'
prediction_points	tuple[str,...]	()

correction
Gt(gt=0.0); Le(le=1.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

prediction_points
MaxLen(max_length=10)

4.1.283 MetaAnalysisResult

tabelz.analysis.meta_analysis_contracts.MetaAnalysisResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MetaAnalysisConfig	required
measure_label	str	required
exponentiate	bool	required
studies	tuple[MetaStudy,...]	required
k	int	required
pooled	tuple[MetaPooled,...]	required
heterogeneity	MetaHeterogeneity	required
subgroups	tuple[MetaSubgroup,...]	()
subgroup_q	float|None	None
subgroup_df	int|None	None
subgroup_p	float|None	None
regression	MetaRegression|None	None
bias	tuple[MetaBias,...]	()
trimfill	MetaTrimFill|None	None
cumulative	tuple[MetaSensitivity,...]	()
leave_one_out	tuple[MetaSensitivity,...]	()
missing_count	int	0
excluded_count	int	0
notes	tuple[str,...]	()

Field	Type	Default
warnings	tuple[str, ...]	()

4.1.284 MetaBias

tabelz.analysis.meta_analysis_contracts.MetaBias

Common strict, immutable model baseline for public contracts.

Field	Type	Default
test	str	required
statistic	float|None	required
p_value	float|None	required
detail	str	required
available	bool	required

4.1.285 MetaCoefficient

tabelz.analysis.meta_analysis_contracts.MetaCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
standard_error	float|None	required
statistic	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required

4.1.286 MetaHeterogeneity

tabelz.analysis.meta_analysis_contracts.MetaHeterogeneity

Common strict, immutable model baseline for public contracts.

Field	Type	Default
q	float	required
df	int	required
p_value	float|None	required
tau2	float|None	required
tau2_lower	float|None	required
tau2_upper	float|None	required

Field	Type	Default
tau2_method	str|None	required
tau2_iterations	int|None	required
tau2_boundary	bool|None	required
i2	float|None	required
i2_lower	float|None	required
i2_upper	float|None	required
h2	float|None	required
h2_lower	float|None	required
h2_upper	float|None	required
hk_scale	float|None	required

4.1.287 MetaPooled

tabelz.analysis.meta_analysis_contracts.MetaPooled

Common strict, immutable model baseline for public contracts.

Field	Type	Default
model	str	required
k	int	required
estimate	float	required
standard_error	float|None	required
statistic	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required
distribution	str	required
prediction_lower	float|None	None
prediction_upper	float|None	None
withheld	str	''

4.1.288 MetaPrediction

tabelz.analysis.meta_analysis_contracts.MetaPrediction

The predicted average effect at one moderator point, and the prediction interval for a new study's true effect there (#143).

Field	Type	Default
point	str	required
estimate	float	required
standard_error	float|None	required
lower	float|None	required

Field	Type	Default
upper	float|None	required
reference_df	int|None	required
prediction_lower	float|None	required
prediction_upper	float|None	required
prediction_df	int|None	required

4.1.289 MetaRegression

tabelz.analysis.meta_analysis_contracts.MetaRegression

Common strict, immutable model baseline for public contracts.

Field	Type	Default
coefficients	tuple[MetaCoefficient, ...]	required
tau2	float|None	required
qm	float|None	required
qm_df	tuple[int, ...]	required
qm_p	float|None	required
qm_kind	str	required
qe	float	required
qe_df	int	required
qe_p	float|None	required
r2	float|None	required
withheld	str	''
predictions	tuple[MetaPrediction, ...]	()

4.1.290 MetaSensitivity

tabelz.analysis.meta_analysis_contracts.MetaSensitivity

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
k	int	required
estimate	float	required
lower	float|None	required
upper	float|None	required
tau2	float|None	required
i2	float|None	required

4.1.291 MetaStudy

tabelz.analysis.meta_analysis_contracts.MetaStudy

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
effect	float|None	required
standard_error	float|None	required
lower	float|None	required
upper	float|None	required
weight_percent	float|None	required
included	bool	required
corrected	bool	required
reason	str	''
subgroup	str|None	None

4.1.292 MetaSubgroup

tabelz.analysis.meta_analysis_contracts.MetaSubgroup

Common strict, immutable model baseline for public contracts.

Field	Type	Default
subgroup	str	required
pooled	MetaPooled	required
tau2	float|None	required
i2	float|None	required
q	float	required

4.1.293 MetaTrimFill

tabelz.analysis.meta_analysis_contracts.MetaTrimFill

Common strict, immutable model baseline for public contracts.

Field	Type	Default
estimator	str	required
side	str	required
missing	int	required
status	str	'completed'
estimate	float	required
lower	float|None	required
upper	float|None	required
filled	tuple[tuple[float,float],...]	required

4.1.294 AnalysisContext

`tabelz.analysis.methods.AnalysisContext`

Strict frozen bundle every analysis method needs to run and report.

Field	Type	Default
`population`	`PopulationRef`	`required`
`input_row_count`	`int`	`required`
`filter`	`NoFilter|SelectionFilter|PredicateFilter`	`factory:NoFilter`
`sources`	`tuple[SourceInput,...]`	`required`
`input_fingerprints`	`tuple[InputFingerprint,...]`	`()`
`operation`	`OperationProvenance`	`required`
`profile`	`ProfileProvenance`	`required`
`engine`	`EngineProvenance`	`required`
`parameters`	`tuple[InvocationParameter,...]`	`()`

input_row_count
Ge(ge=0)

sources
MinLen(min_length=1)

4.1.295 AnalysisOutcome

`tabelz.analysis.methods.AnalysisOutcome`

Typed method outcome: the result envelope plus its reconciled receipt.

Field	Type	Default
`result`	`AnalysisResult`	`required`
`receipt`	`ProvenanceReceipt`	`required`

4.1.296 CrosstabRow

`tabelz.analysis.missing_value_analysis_contracts.CrosstabRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`indicator`	`str`	`required`
`variable`	`str`	`required`
`category`	`str|None`	`required`
`present`	`int`	`required`
`missing`	`int`	`required`
`percent_missing`	`float|None`	`required`

4.1.297 EmIteration

`tabelz.analysis.missing_value_analysis_contracts.EmIteration`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`iteration`	`int`	`required`
`log_likelihood`	`float`	`required`
`max_scaled_change`	`float`	`required`
`log_likelihood_change`	`float`	`required`

4.1.298 EmResult

`tabelz.analysis.missing_value_analysis_contracts.EmResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`status`	`Literal['converged','iterationcapreached','failed']`	`required`
`reason`	`str`	`''`
`tolerance`	`float`	`required`
`max_iterations`	`int`	`required`
`iterations`	`int`	`required`
`cases_used`	`int`	`required`
`all_missing_cases`	`int`	`required`
`means`	`tuple[float,...]`	`()`
`covariance`	`tuple[tuple[float|None,...],...]`	`()`
`correlations`	`tuple[tuple[float|None,...],...]`	`()`
`unidentified_pairs`	`tuple[tuple[str,str],...]`	`()`
`history`	`tuple[EmIteration,...]`	`()`
`log_likelihood`	`float|None`	`None`

4.1.299 ExtremeRow

`tabelz.analysis.missing_value_analysis_contracts.ExtremeRow`

An original row beyond a 1.5-IQR fence, listed once with its multiplicity.

Field	Type	Default
`row_id`	`str`	`required`
`variable`	`str`	`required`
`value`	`float`	`required`
`side`	`Literal['low','high']`	`required`

Field	Type	Default
multiplicity	int	required

4.1.300 LittleTest

tabelz.analysis.missing_value_analysis_contracts.LittleTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
status	Literal['computed','withheld']	required
reason	str	''
statistic	float|None	None
df	int|None	None
p_value	float|None	None
patterns	int	0

4.1.301 MissingValueConfig

tabelz.analysis.missing_value_analysis_contracts.MissingValueConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
quantitative	tuple[str,...]	required
categorical	tuple[str,...]	()
min_percent	float	5.0
adjustment	Literal['holm','none']	'holm'
em_preset	Literal['default','spss','custom']	'default'
em_tolerance	float	1e-06
em_max_iterations	int	100
max_patterns	int	50
weighting	Literal['refuse','weighted_descriptio n']	'refuse'
frequency_weight	str|None	None

quantitative
MinLen(min_length=1); MaxLen(max_length=64)

categorical
MaxLen(max_length=32)

min_percent
Ge(ge=0.0); Le(le=100.0)

em_tolerance
Gt(gt=0.0)

em_max_iterations
Ge(ge=1); Le(le=10000)

max_patterns
Ge(ge=1); Le(le=500)

4.1.302 MissingValueResult

`tabelz.analysis.missing_value_analysis_contracts.MissingValueResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`MissingValueConfig`	`required`
`cases`	`int`	`required`
`excluded_count`	`int`	`required`
`variables`	`tuple[str,...]`	`required`
`univariate`	`tuple[UnivariateRow,...]`	`required`
`patterns`	`tuple[PatternRow,...]`	`required`
`patterns_total`	`int`	`required`
`welch`	`tuple[WelchRow,...]`	`()`
`welch_filtered`	`tuple[tuple[str,float],...]`	`()`
`crosstabs`	`tuple[CrosstabRow,...]`	`()`
`em`	`EmResult`	`required`
`little`	`LittleTest`	`required`
`notes`	`tuple[str,...]`	`()`
`warnings`	`tuple[str,...]`	`()`
`weighting`	`WeightedPopulations|None`	`None`
`extreme_rows`	`tuple[ExtremeRow,...]`	`()`
`extreme_rows_total`	`int`	`0`

4.1.303 PatternRow

`tabelz.analysis.missing_value_analysis_contracts.PatternRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`missing`	`tuple[bool,...]`	`required`
`count`	`int`	`required`
`percent`	`float`	`required`
`variables_missing`	`int`	`required`
`rows`	`int|None`	`None`

4.1.304 UnivariateRow

`tabelz.analysis.missing_value_analysis_contracts.UnivariateRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variable	str	required
kind	Literal['quantitative','categorical']	required
present	int	required
missing	int	required
percent_missing	float	required
mean	float|None	None
sd	float|None	None
low_extremes	int|None	None
high_extremes	int|None	None
present_rows	int|None	None
missing_rows	int|None	None

4.1.305 WeightedPopulations

tabelz.analysis.missing_value_analysis_contracts.WeightedPopulations

The two populations of a weighted description with unweighted inference (#136).

Field	Type	Default
column	str	required
original_rows	int	required
invalid_weight_rows	int	required
zero_weight_rows	int	required
analysed_rows	int	required
weight_total	int	required
unweighted_membership	Literal['positive_weight_rows']	'positive_weight_rows'

4.1.306 WelchRow

tabelz.analysis.missing_value_analysis_contracts.WelchRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
indicator	str	required
variable	str	required
present_count	int	required
missing_count	int	required
present_mean	float|None	required
missing_mean	float|None	required

Field	Type	Default
difference	float|None	required
t_value	float|None	required
df	float|None	required
p_value	float|None	required
adjusted_p_value	float|None	required
status	str	''

4.1.307 MixedModelCoefficient

tabelz.analysis.mixed_model_structures.MixedModelCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
fixed_column	str|None	None
reference_level	str|None	None
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required

4.1.308 MixedModelComponent

tabelz.analysis.mixed_model_structures.MixedModelComponent

One explicitly declared independent random intercept component.

Field	Type	Default
name	str	required
level_column	str	required
relation	Literal['crossed','nested']	required
parent_column	str|None	None

name
MinLen(min_length=1)
level_column
MinLen(min_length=1)

4.1.309 MixedModelPrediction

tabelz.analysis.mixed_model_structures.MixedModelPrediction

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
population_fitted	float	required
conditional_fitted	float	required
observed	float	required
population_residual	float	required
conditional_residual	float	required

4.1.310 MixedModelStructuresConfig

tabelz.analysis.mixed_model_structures.MixedModelStructuresConfig

Replayable limits and roles for the two supported F11 model modes.

Field	Type	Default
response_column	str	required
fixed_numeric_columns	tuple[str,...]	()
fixed_categorical_columns	tuple[str,...]	()
model_mode	Literal['random_slope','variance_components']	'random_slope'
group_column	str|None	None
random_slope_column	str|None	None
components	tuple[MixedModelComponent,...]	()
estimation	Literal['reml','ml']	'reml'
confidence_level	float	0.95
max_rows	int	1000
max_iterations	int	200

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

max_iterations
Ge(ge=1); Le(le=200)

4.1.311 MixedModelStructuresResult

tabelz.analysis.mixed_model_structures.MixedModelStructuresResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method	Literal['models.mixed_model_structures']	'models.mixed_model_structures'
implementation_version	str	'1.0.1'
response_column	str	required
fixed_numeric_columns	tuple[str,...]	required
fixed_categorical_columns	tuple[str,...]	required
model_mode	Literal['random_slope','variance_components']	required
group_column	str|None	None
random_slope_column	str|None	None
components	tuple[MixedModelComponent,...]	()
estimation	Literal['reml','ml']	required
optimizer	Literal['lbfgs','constrained_profile']	'constrained_profile'
max_iterations	int	required
confidence_level	float	0.95
coefficients	tuple[MixedModelCoefficient,...]	required
variance_components	tuple[MixedModelVarianceComponent,...]	required
random_covariance	tuple[tuple[float,...],...]	()
residual_variance	float	required
log_likelihood	float	required
aic	float|None	None
design_columns	tuple[str,...]	required
design_rank	int	required
condition_number	float	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
analyzed_count	int	required
group_count	int	required
component_group_counts	tuple[dict[str,object],...]	()
predictions	tuple[MixedModelPrediction,...]	required
population_fingerprint	str	required
warnings	tuple[str,...]	()
convergence_status	Literal['converged','boundary']	'converged'

Field	Type	Default
fit_search	str	''

design_rank
Ge(ge=1)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
analyzed_count
Ge(ge=1)
group_count
Ge(ge=1)

4.1.312 MixedModelVarianceComponent

tabelz.analysis.mixed_model_structures.MixedModelVarianceComponent

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
boundary	bool	False
indicator_columns	int	required

indicator_columns
Ge(ge=1)

4.1.313 MixedModelCoefficient

tabelz.analysis.mixed_models.MixedModelCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
fixed_column	str|None	None
reference_level	str|None	None
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required

4.1.314 MixedModelConfig

`tabelz.analysis.mixed_models.MixedModelConfig`

Bounded, replayable request for one random-intercept Gaussian model.

Field	Type	Default
`response_column`	`str`	`required`
`group_column`	`str`	`required`
`fixed_numeric_columns`	`tuple[str,...]`	`()`
`fixed_categorical_columns`	`tuple[str,...]`	`()`
`estimation`	`Literal['reml','ml']`	`'reml'`
`confidence_level`	`float`	`0.95`
`max_rows`	`int`	`1000000`
`max_iterations`	`int`	`200`

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

max_iterations
Ge(ge=1); Le(le=200)

4.1.315 MixedModelGroupEffect

`tabelz.analysis.mixed_models.MixedModelGroupEffect`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`group_id`	`str`	`required`
`observations`	`int`	`required`
`residual_sum`	`float`	`required`
`conditional_effect`	`float`	`required`

observations
Ge(ge=1)

4.1.316 MixedModelPrediction

`tabelz.analysis.mixed_models.MixedModelPrediction`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
group_id	str	required
observed	float	required
population_fitted	float	required
conditional_fitted	float	required
population_residual	float	required
conditional_residual	float	required

4.1.317 MixedModelResult

tabelz.analysis.mixed_models.MixedModelResult

Strict numerical result suitable for worker transport and report replay.

Field	Type	Default
method	Literal['models.mixed_linear_random_intercept']	'models.mixed_linear_random_intercept'
implementation_version	str	'1.0.1'
response_column	str	required
group_column	str	required
fixed_numeric_columns	tuple[str,...]	required
fixed_categorical_columns	tuple[str,...]	required
estimation	Literal['reml','ml']	required
optimizer	Literal['lbfgs','profile_search','balanced_closed_form']	'profile_search'
max_iterations	int	required
confidence_level	float	required
coefficients	tuple[MixedModelCoefficient,...]	required
group_effects	tuple[MixedModelGroupEffect,...]	required
predictions	tuple[MixedModelPrediction,...]	required
group_variance	float	required
residual_variance	float	required
icc	float	required
convergence_status	Literal['converged','boundary']	required
conditioning_number	float	required
design_rank	int	required
design_columns	tuple[str,...]	required
source_count	int	required

Field	Type	Default
excluded_count	int	required
missing_count	int	required
analyzed_count	int	required
group_count	int	required
group_counts	tuple[dict[str,object],...]	required
population_fingerprint	str	required
warnings	tuple[str,...]	()
fit_search	str	''

design_rank
Ge(ge=1)

source_count
Ge(ge=0)

excluded_count
Ge(ge=0)

missing_count
Ge(ge=0)

analyzed_count
Ge(ge=1)

group_count
Ge(ge=2)

4.1.318 ClusterProfile

`tabelz.analysis.model_based_clustering.ClusterProfile`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
cluster	int	required
weight	float	required
size	int	required
means	tuple[float,...]	()
variances	tuple[float,...]	()
probabilities	tuple[tuple[tuple[str,float],...],...]	()

4.1.319 KFit

`tabelz.analysis.model_based_clustering.KFit`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
k	int	required
status	Literal['fitted','nosuccessfulfit']	required
reason	str	''
log_likelihood	float|None	None
parameters	int	required
bic	float|None	None
successful_starts	int	0
unconverged_starts	int	0
failed_starts	int	0

4.1.320 ModelClusteringArtifact

tabelz.analysis.model_based_clustering.ModelClusteringArtifact

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	str	required
covariance	str	required
variables	tuple[str,...]	required
weights	tuple[float,...]	required
means	tuple[tuple[float,...],...]	()
covariances	tuple[tuple[tuple[float,...],...],..]	()
categories	tuple[tuple[str,...],...]	()
probabilities	tuple[tuple[tuple[float,...],...],..]	()
missing_items	Literal['observed']|None	None
numeric_variables	tuple[str,...]|None	None
nominal_variables	tuple[str,...]|None	None
fit_mean	tuple[float,...]|None	None
fit_sd	tuple[float,...]|None	None
global_variance	tuple[float,...]|None	None

4.1.321 ModelClusteringConfig

tabelz.analysis.model_based_clustering.ModelClusteringConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variables	tuple[str,...]	required

Field	Type	Default
kind	Literal['gaussian','latent_class','two_step']	'gaussian'
covariance	Literal['diagonal','full']	'diagonal'
k_min	int	1
k_max	int	6
seed	int	20260928
missing_items	Literal['complete','observed']|None	None
silhouette_sample_size	int|None	None
cf_threshold	float|None	None
cf_max_leaves	int|None	None

variables
MinLen(min_length=1); MaxLen(max_length=32)
k_min
Ge(ge=1); Le(le=20)
k_max
Ge(ge=1); Le(le=20)
seed
Ge(ge=0)
silhouette_sample_size
Ge(ge=2); Le(le=5000)
cf_threshold
Gt(gt=0)
cf_max_leaves
Ge(ge=8); Le(le=128)

4.1.322 ModelClusteringResult

tabelz.analysis.model_based_clustering.ModelClusteringResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ModelClusteringConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
fits	tuple[KFit,...]	required
chosen_k	int	required
profiles	tuple[ClusterProfile,...]	required
silhouette	float|None	required
silhouette_note	str	required

Field	Type	Default
artifact	ModelClusteringArtifact	required
notes	tuple[str,...]	()
all_missing_count	int|None	None
silhouette_sample_n	int|None	None
silhouette_standard_error	float|None	None

all_missing_count
Ge(ge=0)
silhouette_sample_n
Ge(ge=2)
silhouette_standard_error
Ge(ge=0)

4.1.323 DesirabilityGoal

tabelz.analysis.model_optimization.DesirabilityGoal

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_name	str	required
goal	Literal['maximize','minimize','target']	required
weight	float	1.0
low	float	required
high	float	required
target	float|None	None

response_name
MinLen(min_length=1)
weight
Gt(gt=0.0)

4.1.324 PredictorSpecification

tabelz.analysis.model_optimization.PredictorSpecification

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
kind	Literal['numeric','categorical']	required
min_val	float|None	None
max_val	float|None	None

Field	Type	Default
categories	tuple[str,...]|None	None
current_value	float|str	required

name
MinLen(min_length=1)

4.1.325 ModelScoringArtifact

tabelz.analysis.model_scoring.ModelScoringArtifact

Safe, immutable JSON model snapshot; never a pickled estimator.

Field	Type	Default
schema_version	str	'1.0.0'
implementation_version	str	'1.0.1'
method	str	'model_scoring.frozen_regression'
family	Literal['ols','wls','logistic','poisson']	required
response	str	required
predictors	tuple[ModelScoringPredictor,...]	required
feature_order	tuple[str,...]	required
coefficient_names	tuple[str,...]	required
coefficients	tuple[float,...]	required
covariance	tuple[tuple[float,...],...]	required
covariance_factor	tuple[tuple[float,...],...]|None	None
link	Literal['identity','logit','log']	required
event_value	str|None	None
reference_value	str|None	None
exposure_column	str|None	None
weight_column	str|None	None
interval_kind	Literal['mean','individual']	'mean'
confidence_level	float	0.95
residual_variance	float|None	None
residual_df	int|None	None
design_rank	int	required
fitted_row_ids	tuple[str,...]	required
provenance	dict[str,str]	factory:dict

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

fitted_row_ids
MinLen(min_length=1); MaxLen(max_length=1000000)

4.1.326 ModelScoringConfig

`tabelz.analysis.model_scoring.ModelScoringConfig`

Explicit fitting settings shared by all supported regression families.

Field	Type	Default
`family`	`Literal['ols','wls','logistic','poisson']`	required
`response`	`str`	required
`predictors`	`tuple[str,...]`	required
`weight_column`	`str|None`	None
`exposure_column`	`str|None`	None
`event_value`	`str|None`	None
`interval_kind`	`Literal['mean','individual']`	'mean'
`confidence_level`	`float`	0.95
`max_rows`	`int`	1000000

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.327 ModelScoringInfluence

`tabelz.analysis.model_scoring.ModelScoringInfluence`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`row_id`	`str`	required
`leverage`	`float|None`	required
`studentized_residual`	`float|None`	required
`cooks_distance`	`float|None`	required
`dfbetas`	`tuple[float|None,...]`	required
`residual`	`float|None`	None
`deleted_studentized_residual`	`float|None`	None
`dffits`	`float|None`	None

4.1.328 ModelScoringPredictor

`tabelz.analysis.model_scoring.ModelScoringPredictor`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
kind	Literal['numeric','categorical']	required
levels	tuple[str,...]	()
reference_level	str|None	None
transform	str	'identity'

4.1.329 ModelScoringResult

tabelz.analysis.model_scoring.ModelScoringResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
artifact	ModelScoringArtifact	required
row_ids	tuple[str,...]	required
predictions	tuple[float,...]	required
lower	tuple[float|None,...]	()
upper	tuple[float|None,...]	()
missing_row_ids	tuple[str,...]	()
excluded_row_ids	tuple[str,...]	()
influence	tuple[ModelScoringInfluence,...]	()
influence_eligible	bool	False
influence_note	str|None	None
warnings	tuple[str,...]	()
source_count	int	required

4.1.330 CategoricalSimulationSummary

tabelz.analysis.model_simulation_contracts.CategoricalSimulationSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
levels	tuple[str,...]	required
probabilities	tuple[float,...]	required
counts	tuple[int,...]	required
note	str	required

4.1.331 CorrelationRow

tabelz.analysis.model_simulation_contracts.CorrelationRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
first	str	required
second	str	required
target_spearman	float	required
latent_pearson	float	required
achieved_spearman	float|None	required

4.1.332 FittedInput

tabelz.analysis.model_simulation_contracts.FittedInput

Input fit; fixed by default, with explicit opt-in transformed uncertainty.

Field	Type	Default
name	str	required
column	str	required
family	str	required
parameters	tuple[NamedParameter,...]	required
observations	int	required
missing	int	required
excluded	int	required
method	str	required
parameter_uncertainty	FittedParameterUncertainty|None	None

4.1.333 FittedParameterUncertainty

tabelz.analysis.model_simulation_contracts.FittedParameterUncertainty

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method	Literal['expected-information-transformed-normal-v1']	required
coordinates	tuple[str,...]	required
center	tuple[float,...]	required
covariance	tuple[tuple[float,...],...]	required
note	str	required

4.1.334 HistogramBin

tabelz.analysis.model_simulation_contracts.HistogramBin

Common strict, immutable model baseline for public contracts.

Field	Type	Default
lower	float	required
upper	float	required
count	int	required

4.1.335 InputSummary

tabelz.analysis.model_simulation_contracts.InputSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
role	Literal['random','fixed']	required
source	Literal['entered','fitted','fixed']	required
units	str	required
distribution	DistributionSummary|None	required
fixed_value	float|None	required
fixed_level	str|None	required
sample_mean	float|None	required
sample_standard_deviation	float|None	required
minimum	float|None	required
maximum	float|None	required
spearman_with_target	float|None	required
spearman_note	str	required
outside_fitted_range	float|None	required

4.1.336 ModelSimulationConfig

tabelz.analysis.model_simulation_contracts.ModelSimulationConfig

A saved simulation plan: model, inputs, dependence, draws, seed and summaries.

Field	Type	Default
model_kind	Literal['stored','formula']	'stored'
model	str|None	None
stored_model	ModelScoringArtifact|None	None
stored_model_title	str|None	None
formula	str|None	None
target_name	str	'Y'
target_units	str	''

Field	Type	Default
target	Literal['mean','outcome']	'mean'
inputs	tuple[SimulationInput,...]	required
correlations	tuple[SpearmanTarget,...]	()
draws	int	10000
seed	int	20260928
thresholds	tuple[float,...]	()
percentiles	tuple[float,...]	(1.0,5.0,10.0,25.0,50.0,75.0,90.0,95.0,99.0)
confidence_level	float	0.95
rng_method	Literal['pcg64-inverse-cdf-gaussian-copula-v1']	'pcg64-inverse-cdf-gaussian-copula-v1'
input_parameter_uncertainty	Literal['independent_asymptotic']|None	None
coefficient_uncertainty	Literal['asymptotic_normal']|None	None
sampling	Literal['lhs','iman_conover']|None	None
lhs_replicates	int|None	None
split_streams	Literal['common','independent']|None	None
split_stream	SimulationSplitStream|None	None
standardized_sensitivity	bool|None	None
one_sided_bound	Literal['upper','lower']|None	None

stored_model_title
MaxLen(max_length=200)

formula
MaxLen(max_length=2048)

target_units
MaxLen(max_length=40)

inputs
MinLen(min_length=1); MaxLen(max_length=32)

draws
Ge(ge=100); Le(le=1000000)

seed
Ge(ge=0); Le(le=9223372036854775807)

thresholds
MaxLen(max_length=20)

percentiles
MaxLen(max_length=25)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

lhs_replicates
Ge(ge=2); Le(le=100)

4.1.337 ModelSimulationResult

`tabelz.analysis.model_simulation_contracts.ModelSimulationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method_identity	str	required
implementation_version	str	required
rng_method	str	required
rng_description	str	required
numpy_version	str	required
scipy_version	str	required
config	ModelSimulationConfig	required
model_label	str	required
target_label	str	required
target_description	str	required
draws	int	required
mean	float	required
standard_deviation	float	required
mean_monte_carlo_standard_error	float	required
minimum	float	required
maximum	float	required
percentiles	tuple[PercentileRow,...]	required
thresholds	tuple[ThresholdRow,...]	required
histogram	tuple[HistogramBin,...]	required
inputs	tuple[InputSummary,...]	required
correlations	tuple[CorrelationRow,...]	required
latent_smallest_eigenvalue	float|None	required
fitted_inputs	tuple[FittedInput,...]	required
warnings	tuple[str,...]	required
notes	tuple[str,...]	required
output_fingerprint	str	required
replicated_precision	ReplicatedSimulationPrecision|None	None
categorical_inputs	tuple[CategoricalSimulationSummary,...]|None	None
standardized_sensitivity	StandardizedSensitivity|None	None

Field	Type	Default
one_sided_thresholds	tuple[ThresholdRow,...]|None	None

4.1.338 PercentileRow

tabelz.analysis.model_simulation_contracts.PercentileRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
percent	float	required
value	float|None	required
tail_draws	float	required
note	str	required

4.1.339 ReplicatedSimulationPrecision

tabelz.analysis.model_simulation_contracts.ReplicatedSimulationPrecision

Common strict, immutable model baseline for public contracts.

Field	Type	Default
method	Literal['randomized-lhs-seed-sequence-v1','lhs-iman-conover-normal-scores-v1']	required
replicate_seeds	tuple[int,...]	required
replicate_means	tuple[float,...]	required
percentile_standard_errors	tuple[float,...]	required
replicate_achieved_correlations	tuple[tuple[float|None,...],...]	required
note	str	required

4.1.340 SimulationInput

tabelz.analysis.model_simulation_contracts.SimulationInput

One model input: a random draw (entered or fitted parameters) or a fixed value.

Field	Type	Default
name	str	required
source	Literal['entered','fitted','fixed','categorical']	required
probabilities	dict[str,float]|None	None

Field	Type	Default
distribution	NormalDistribution|StudentTDistribution|ChiSquareDistribution|FDistribution|BinomialDistribution|PoissonDistribution|ExponentialDistribution|WeibullDistribution|None	None
fit_family	Literal['normal','t','chi_square','f','binomial','poisson','exponential','weibull']|None	None
fit_column	str|None	None
fit_trials	int|None	None
value	float|None	None
level	str|None	None
units	str	''

fit_trials
Ge(ge=1); Le(le=1000000000)

units
MaxLen(max_length=40)

4.1.341 SimulationSplitStream

`tabelz.analysis.model_simulation_contracts.SimulationSplitStream`

Recorded ordered group-to-child mapping of an independent split-file run.

Field	Type	Default
ordered_group_keys	tuple[str,...]	required
group_index	int	required
method	Literal['seed-sequence-spawn-v1']	'seed-sequence-spawn-v1'

ordered_group_keys
MinLen(min_length=1); MaxLen(max_length=50)

group_index
Ge(ge=0); Le(le=49)

4.1.342 SpearmanTarget

`tabelz.analysis.model_simulation_contracts.SpearmanTarget`

A population Spearman correlation between two continuous random inputs.

Field	Type	Default
first	str	required
second	str	required

Field	Type	Default
spearman	float	required

4.1.343 StandardizedSensitivity

tabelz.analysis.model_simulation_contracts.StandardizedSensitivity

Common strict, immutable model baseline for public contracts.

Field	Type	Default
rows	tuple[StandardizedSensitivityRow, ...]	required
r_squared	float|None	required
design_rank	int	required
active_inputs	int	required
note	str	required

4.1.344 StandardizedSensitivityRow

tabelz.analysis.model_simulation_contracts.StandardizedSensitivityRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
input	str	required
coefficient	float|None	required
note	str	required

4.1.345 ThresholdRow

tabelz.analysis.model_simulation_contracts.ThresholdRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
threshold	float	required
exceedances	int	required
probability	float|None	required
monte_carlo_standard_error	float|None	required
lower	float|None	required
upper	float|None	required
bound	Literal['two_sided','upper','lower', 'unavailable']	'two_sided'
note	str	''

4.1.346 MdsConfig

`tabelz.analysis.multidimensional_scaling.MdsConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`source`	`Literal['variables','matrix']`	`'variables'`
`columns`	`tuple[str,...]`	required
`labels`	`str|None`	`None`
`standardize`	`bool`	`True`
`standardize_for_correlation`	`bool`	`False`
`distance`	`Literal['euclidean','cityblock','correlation']`	`'euclidean'`
`objects`	`Literal['rows','variables']`	`'rows'`
`model`	`Literal['classical','ordinal']`	`'classical'`
`dimensions`	`int`	`2`
`ties`	`Literal['secondary','primary']`	`'secondary'`
`extra_starts`	`int`	`0`
`seed`	`int`	`20260928`

columns
MinLen(min_length=1)
dimensions
Ge(ge=1); Le(le=6)
extra_starts
Ge(ge=0); Le(le=50)
seed
Ge(ge=0)

4.1.347 MdsEigenvalue

`tabelz.analysis.multidimensional_scaling.MdsEigenvalue`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`dimension`	`int`	required
`eigenvalue`	`float`	required
`retained`	`bool`	required

4.1.348 MdsObject

`tabelz.analysis.multidimensional_scaling.MdsObject`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
coordinates	tuple[float, ...]	required

4.1.349 MdsPair

tabelz.analysis.multidimensional_scaling.MdsPair

Common strict, immutable model baseline for public contracts.

Field	Type	Default
dissimilarity	float	required
distance	float	required
disparity	float|None	None

4.1.350 MdsResult

tabelz.analysis.multidimensional_scaling.MdsResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MdsConfig	required
objects	tuple[MdsObject, ...]	required
dimensions	int	required
rows_used	int	required
missing_count	int	required
excluded_count	int	required
eigenvalues	tuple[MdsEigenvalue, ...]	()
gof_absolute	float|None	None
gof_positive	float|None	None
starts	tuple[MdsStart, ...]	()
iterations	int|None	None
converged	bool|None	None
raw_stress	float|None	None
normalized_stress	float|None	None
smacof_stress	float|None	None
stress_1	float|None	None
shepard	tuple[MdsPair, ...]	()
notes	tuple[str, ...]	()
warnings	tuple[str, ...]	()

4.1.351 MdsStart

tabelz.analysis.multidimensional_scaling.MdsStart

Common strict, immutable model baseline for public contracts.

Field	Type	Default
start	str	required
iterations	int	required
converged	bool	required
normalized_stress	float	required

4.1.352 ComparisonResult

tabelz.analysis.multiple_comparisons.ComparisonResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
procedure	str	required
name	str	required
applicability	Literal['independentpooled','independentunequal','correlated']	required
alpha	float	required
error_term	str	required
critical_value	float|None	required
rows	tuple[PairwiseRow,...]	required
subsets	tuple[tuple[str,...],...]	()
notes	tuple[str,...]	()
critical_bracket	tuple[float,float]|None	None
calibration	MaxTCalibration|None	None

4.1.353 GroupSummary

tabelz.analysis.multiple_comparisons.GroupSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
n	int	required
mean	float	required
variance	float|None	None

n
Ge(ge=1)

4.1.354 PairwiseRow

`tabelz.analysis.multiple_comparisons.PairwiseRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`first`	`str`	`required`
`second`	`str`	`required`
`difference`	`float`	`required`
`standard_error`	`float`	`required`
`statistic`	`float`	`required`
`df`	`float|None`	`required`
`p_adjusted`	`float|None`	`required`
`lower`	`float|None`	`required`
`upper`	`float|None`	`required`
`significant`	`bool`	`required`
`p_half_width`	`float|None`	`None`
`indeterminate`	`bool`	`False`
`p_note`	`str`	`''`

4.1.355 McaCategory

`tabelz.analysis.multiple_correspondence_analysis.McaCategory`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`variable`	`str`	`required`
`category`	`str`	`required`
`is_missing`	`bool`	`False`
`count`	`int`	`required`
`mass`	`float|None`	`required`
`coordinates`	`tuple[float,...]`	`required`
`contributions`	`tuple[float,...]`	`()`
`cosine_squared`	`tuple[float,...]`	`()`
`supplementary`	`bool`	`False`

4.1.356 McaConfig

`tabelz.analysis.multiple_correspondence_analysis.McaConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
columns	tuple[str,...]	required
supplementary	tuple[str,...]	()
missing	Literal['complete','category']	'complete'
dimensions	int	2

columns
MinLen(min_length=2); MaxLen(max_length=20)

dimensions
Ge(ge=1); Le(le=10)

4.1.357 McaDimension

tabelz.analysis.multiple_correspondence_analysis.McaDimension

Common strict, immutable model baseline for public contracts.

Field	Type	Default
dimension	int	required
inertia	float	required
raw_percent	float	required
cumulative_percent	float	required
adjusted	float	required
benzecri_percent	float|None	required
greenacre_percent	float|None	required

4.1.358 McaResult

tabelz.analysis.multiple_correspondence_analysis.McaResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	McaConfig	required
rows_analysed	int	required
excluded_count	int	required
incomplete_count	int	required
incomplete_row_ids	tuple[str,...]	required
variables	int	required
categories	int	required
total_inertia	float	required
greenacre_denominator	float|None	required
dimensions	tuple[McaDimension,...]	required

Field	Type	Default
categories_table	tuple[McaCategory, ...]	required
removed_categories	tuple[str, ...]	()
warnings	tuple[str, ...]	()
truncated_inertia	float	0.0
smallest_eigenvalue	float	0.0

4.1.359 MrCategory

tabelz.analysis.multiple_response.MrCategory

Common strict, immutable model baseline for public contracts.

Field	Type	Default
category	str	required
label	str	required
responses	int	required
percent_of_responses	float|None	required
percent_of_cases	float|None	required
cases	int	0

4.1.360 MrCell

tabelz.analysis.multiple_response.MrCell

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row	str	required
row_label	str	required
column	str	required
column_label	str	required
count	int	required
row_percent	float|None	required
column_percent	float|None	required
total_percent	float|None	required
cases	int	0
responses	int	0

4.1.361 MrCrosstabConfig

tabelz.analysis.multiple_response.MrCrosstabConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
set_name	str	required
response_set	ResponseSet	required
by_column	str|None	None
by_set_name	str|None	None
by_set	ResponseSet|None	None
base	Literal['cases','responses']	'cases'
category_labels	dict[str,str]	factory:dict
column_labels	dict[str,str]	factory:dict

set_name
MinLen(min_length=1)

4.1.362 MrCrosstabResult

tabelz.analysis.multiple_response.MrCrosstabResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MrCrosstabConfig	required
rows_analysed	int	required
excluded_count	int	required
valid_cases	int	required
left_out_cases	int	required
base_total	int	required
row_totals	dict[str,int]	required
column_totals	dict[str,int]	required
cells	tuple[MrCell,...]	required
note	str	'Nochi-squaretestisgiven:acasewithseveralresponsescountsinseveralcells,sothecellsarenotindependent.'

4.1.363 MrFrequenciesConfig

tabelz.analysis.multiple_response.MrFrequenciesConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
set_name	str	required
response_set	ResponseSet	required

Field	Type	Default
category_labels	dict[str,str]	factory:dict

set_name
MinLen(min_length=1)

4.1.364 MrFrequenciesResult

tabelz.analysis.multiple_response.MrFrequenciesResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	MrFrequenciesConfig	required
rows_analysed	int	required
excluded_count	int	required
valid_cases	int	required
cases_without_response	int	required
all_missing_rows	int	required
total_responses	int	required
categories	tuple[MrCategory,...]	required

4.1.365 ClassRow

tabelz.analysis.nearest_neighbour.ClassRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
observed	str	required
counts	tuple[int,...]	required

4.1.366 FeatureImportance

tabelz.analysis.nearest_neighbour.FeatureImportance

Common strict, immutable model baseline for public contracts.

Field	Type	Default
feature	str	required
reduced_validation_error	float	required
change	float	required

4.1.367 KChoice

`tabelz.analysis.nearest_neighbour.KChoice`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`k`	`int`	`required`
`validation_error`	`float`	`required`

4.1.368 NearestNeighbourArtifact

`tabelz.analysis.nearest_neighbour.NearestNeighbourArtifact`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`task`	`str`	`required`
`distance`	`str`	`required`
`k`	`int`	`required`
`target`	`str`	`''`
`features`	`tuple[str,...]`	`required`
`means`	`tuple[float,...]`	`required`
`scales`	`tuple[float,...]`	`required`
`training`	`tuple[tuple[float,...],...]`	`required`
`targets`	`tuple[float,...]`	`required`
`classes`	`tuple[str,...]`	`()`
`training_row_ids`	`tuple[str,...]`	`()`
`validation_row_ids`	`tuple[str,...]`	`()`
`test_row_ids`	`tuple[str,...]`	`()`
`large_exact`	`bool|None`	`None`

4.1.369 NearestNeighbourConfig

`tabelz.analysis.nearest_neighbour.NearestNeighbourConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`target`	`str`	`required`
`features`	`tuple[str,...]`	`required`
`task`	`Literal['classification','regression']`	`'classification'`
`class_order`	`tuple[str,...]`	`()`
`distance`	`Literal['euclidean','cityblock']`	`'euclidean'`
`k_values`	`tuple[int,...]`	`(1,3,5,7,9,11,15)`

Field	Type	Default
test_fraction	float	0.2
validation_fraction	float	0.25
seed	int	20260928
large_exact	bool|None	None

features
MinLen(min_length=1); MaxLen(max_length=64)

class_order
MaxLen(max_length=200)

test_fraction
Ge(ge=0.1); Le(le=0.5)

validation_fraction
Ge(ge=0.1); Le(le=0.5)

seed
Ge(ge=0)

4.1.370 NearestNeighbourResult

tabelz.analysis.nearest_neighbour.NearestNeighbourResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	NearestNeighbourConfig	required
training_rows	int	required
validation_rows	int	required
test_rows	int	required
missing_count	int	required
excluded_count	int	required
grid	tuple[KChoice, ...]	required
chosen_k	int	required
validation_error	float	required
test_error	float	required
error_name	str	required
importance	tuple[FeatureImportance, ...]	required
classes	tuple[str, ...]	()
confusion	tuple[ClassRow, ...]	()
artifact	NearestNeighbourArtifact	required
notes	tuple[str, ...]	()

4.1.371 NonlinearParameter

tabelz.analysis.nonlinear_contracts.NonlinearParameter

!!! abstract ”Usage Documentation” [Models](../concepts/models.md)

Field	Type	Default
name	str	required
start	float	required
lower	float|None	None
upper	float|None	None

4.1.372 NonlinearRegressionArtifact

tabelz.analysis.nonlinear_contracts.NonlinearRegressionArtifact

!!! abstract ”Usage Documentation” [Models](../concepts/models.md)

Field	Type	Default
artifact_version	Literal[1]	1
implementation	Literal['nonlinear_regression.least_squares.v1']	'nonlinear_regression.least_squares.v1'
config	NonlinearRegressionConfig	required
estimates	tuple[float,...]	required
covariance	tuple[tuple[float,...],...]|None	required
residual_variance	float	required
degrees_of_freedom	int	required
fit_x_range	tuple[float,float]	required
provenance	dict[str,str|int|float|bool|None]	factory:dict

residual_variance
Ge(ge=0)

degrees_of_freedom
Gt(gt=0)

4.1.373 NonlinearRegressionConfig

tabelz.analysis.nonlinear_contracts.NonlinearRegressionConfig

!!! abstract ”Usage Documentation” [Models](../concepts/models.md)

Field	Type	Default
x_column	str	required
response	str	required
curve	Literal['exponential_decay','asymptotic_growth','four_parameter_logistic','custom']	'exponential_decay'
expression	str	''
parameters	tuple[NonlinearParameter,...]	required

Field	Type	Default
confidence_level	float	0.95
max_evaluations	int	1000
max_rows	int	1000000

x_column
MinLen(min_length=1)
response
MinLen(min_length=1)
parameters
MinLen(min_length=1); MaxLen(max_length=8)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
max_evaluations
Ge(ge=1); Le(le=10000)
max_rows
Ge(ge=2); Le(le=1000000)

4.1.374 NonlinearRegressionResult

tabelz.analysis.nonlinear_contracts.NonlinearRegressionResult

!!! abstract "Usage Documentation" [Models](../concepts/models.md)

Field	Type	Default
artifact	NonlinearRegressionArtifact	required
mode	Literal['fit','predict']	required
rows	tuple[NonlinearRow,...]	required
eligible_count	int	required
missing_count	int	required
excluded_count	int	required
sse	float|None	None
rmse	float|None	None
rank	int|None	None
condition	float|None	None
evaluations	int	0
model_evaluations	int	0
convergence	str	'Retainedfit;nooptimizationperformed.'
parameter_intervals	tuple[tuple[float,float]|None,...]	()
warnings	tuple[str,...]	()

4.1.375 NonlinearRow

tabelz.analysis.nonlinear_contracts.NonlinearRow

!!! abstract "Usage Documentation" [Models](../concepts/models.md)

Field	Type	Default
row_id	str	required
x	float|None	required
observed	float|None	None
predicted	float|None	None
residual	float|None	None
mean_lower	float|None	None
mean_upper	float|None	None
status	Literal['eligible','missing','excluded']	required

4.1.376 PanelFixedEffectsCoefficient

tabelz.analysis.panel_fixed_effects.PanelFixedEffectsCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
predictor	str	required
estimate	float	required
standard_error	float	required
t_statistic	float	required
p_value	float	required
degrees_of_freedom	int	required
lower	float	required
upper	float	required

4.1.377 PanelFixedEffectsConfig

tabelz.analysis.panel_fixed_effects.PanelFixedEffectsConfig

Strict, immutable bounded panel fixed-effects request.

Field	Type	Default
response_column	str	required
predictor_columns	tuple[str,...]	required
entity_column	str	required
time_column	str	required
confidence_level	float	0.95
max_rows	int	1000000

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.378 PanelFixedEffectsObservation

`tabelz.analysis.panel_fixed_effects.PanelFixedEffectsObservation`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
entity	str|None	required
time	str|None	required
response	float|None	required
fitted	float|None	required
residual	float|None	required
included	bool	required
omission_reason	str|None	None

4.1.379 PanelFixedEffectsResult

`tabelz.analysis.panel_fixed_effects.PanelFixedEffectsResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_column	str	required
predictor_columns	tuple[str,...]	required
entity_column	str	required
time_column	str	required
confidence_level	float	required
max_rows	int	required
method	str	'panel.fixed_effects.within_entity_cr1'
implementation_version	str	'1.0.0'
observations	int	required
source_count	int	required
missing_count	int	required
excluded_count	int	required
singleton_count	int	required
singleton_observation_count	int	required
entity_count	int	required
time_count	int	required

Field	Type	Default
entity_counts	dict[str,int]	required
time_counts	dict[str,int]	required
entity_intercepts	dict[str,float]	required
entity_coverage	tuple[dict[str,object],...]	()
invariant_predictors	tuple[str,...]	()
design_rank	int	required
covariance_df	int	required
residual_df	int	required
parameter_count	int	required
reference_df	int	required
covariance_correction	str	required
within_r_squared	float|None	required
coefficients	tuple[PanelFixedEffectsCoefficient,...]	required
row_results	tuple[PanelFixedEffectsObservation,...]	required
retained_row_ids	tuple[str,...]	required
population_fingerprint	str	required
warnings	tuple[str,...]	()

4.1.380 ParetoConfig

tabelz.analysis.pareto.ParetoConfig

Field	Type	Default
category	str	required
frequency	str|None	None
max_categories	int	20
combine_remainder	bool	True

4.1.381 ParetoResult

tabelz.analysis.pareto.ParetoResult

Field	Type	Default
config	ParetoConfig	required
source_rows	int	required
complete_rows	int	required
dropped_rows	int	required

Field	Type	Default
category_count	int	required
total_frequency	float	required
rows	tuple[ParetoRow, ...]	required
combined_count	int	required
omitted_count	int	required

4.1.382 ParetoRow

tabelz.analysis.pareto.ParetoRow

Field	Type	Default
category	str	required
frequency	float	required
percent	float	required
cumulative_percent	float	required
source_categories	tuple[str, ...]	required

4.1.383 PermutationNullBin

tabelz.analysis.permutation_tests.PermutationNullBin

One fixed-width interval of the permutation null distribution.

Field	Type	Default
index	int	required
lower	float	required
upper	float	required
count	int	required

4.1.384 PermutationTestsConfig

tabelz.analysis.permutation_tests.PermutationTestsConfig

Strict request for one permutation test.

Field	Type	Default
mode	Literal['independent','paired']	required
response_column	str	required
group_column	str|None	None
group_a	str|None	None
group_b	str|None	None
second_response_column	str|None	None

Field	Type	Default
alternative	Literal['two-sided','less','greater']	'two-sided'
inference	Literal['exact','monte_carlo']	'monte_carlo'
resamples	int	2000
seed	int	0
histogram_bins	int	40
max_rows	int	1000000
max_cells	int	20000000

resamples
Ge(ge=2); Le(le=10000)

seed
Ge(ge=0); Le(le=9223372036854775807)

histogram_bins
Ge(ge=1); Le(le=64)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

4.1.385 PermutationTestsResult

tabelz.analysis.permutation_tests.PermutationTestsResult

Statistic, tail probability, and the exact input partition.

Field	Type	Default
mode	Literal['independent','paired']	required
alternative	Literal['two-sided','less','greater']	required
inference	Literal['exact','monte_carlo']	required
response_column	str	required
group_column	str|None	required
group_a	str|None	required
group_b	str|None	required
second_response_column	str|None	required
statistic	float	required
p_value	float	required
assignment_count	int	required
seed	int|None	None
source_count	int	required
analyzed_count	int	required
missing_count	int	required
excluded_count	int	required

Field	Type	Default
outside_group_count	int	0
retained_row_ids	tuple[str,...]	()
group_a_row_ids	tuple[str,...]	()
group_b_row_ids	tuple[str,...]	()
group_summaries	tuple[dict[str,object],...]	()
population_fingerprint	str	required
null_histogram_bins	tuple[PermutationNullBin,...]	()
warnings	tuple[str,...]	()
method	str	'permutation_tests.mean_difference'
implementation_version	str	'1.0.0'

4.1.386 DensityPoint

tabelz.analysis.plot_contracts.DensityPoint

Field	Type	Default
value	float	required
density	float	required

4.1.387 DistributionData

tabelz.analysis.plot_contracts.DistributionData

Field	Type	Default
y_column	str	required
group_column	str|None	required
groups	tuple[DistributionGroup,...]	required
source_count	int	required
valid_count	int	required
sampled	bool	required
group_columns	tuple[str,...]	()
group_values	tuple[tuple[str,tuple[str,...]],...]	()

4.1.388 DistributionGroup

tabelz.analysis.plot_contracts.DistributionGroup

One bounded group with exact summaries and sampled drawable points.

Field	Type	Default
label	str	required
valid_count	int	required
minimum	float	required
q1	float	required
median	float	required
q3	float	required
maximum	float	required
whisker_low	float	required
whisker_high	float	required
samples	tuple[DistributionPoint,...]	required
density	tuple[DensityPoint,...]	required
bins	tuple[HistogramBin,...]	required
split_labels	tuple[str,...]	()

4.1.389 DistributionPoint

tabelz.analysis.plot_contracts.DistributionPoint

Field	Type	Default
value	float	required
row_id	str	required

4.1.390 FitPoint

tabelz.analysis.plot_contracts.FitPoint

Field	Type	Default
x	float	required
y	float	required
lower	float	required
upper	float	required

4.1.391 HistogramBin

tabelz.analysis.plot_contracts.HistogramBin

One numeric interval and its aggregated source-row count.

Field	Type	Default
index	int	required
lower	float	required
upper	float	required

Field	Type	Default
count	int	required

4.1.392 HistogramData

tabelz.analysis.plot_contracts.HistogramData

Bounded histogram aggregation independent of source row count.

Field	Type	Default
column	str	required
bins	tuple[HistogramBin,...]	required
source_count	int	required
valid_count	int	required
minimum	float|None	required
maximum	float|None	required

4.1.393 LinearFitData

tabelz.analysis.plot_contracts.LinearFitData

All-row ordinary-least-squares line with a mean-response uncertainty band.

Field	Type	Default
points	tuple[FitPoint,...]	required
slope	float	required
intercept	float	required
r_squared	float	required
valid_count	int	required

4.1.394 PlotData

tabelz.analysis.plot_contracts.PlotData

Bounded scatter payload with explicit source/rendered counts.

Field	Type	Default
frame	polars.dataframe.frame.DataFrame	required
sampled	bool	required
source_count	int	required
seed	int|None	None
valid_count	int|None	None

4.1.395 PrecisionPlanningConfig

`tabelz.analysis.precision_planning.PrecisionPlanningConfig`

Strict, replayable request for a bounded precision plan.

Field	Type	Default
mode	Literal['mean','proportion','normal_tolerance','distribution_free_tolerance']	'mean'
criterion	Literal['absolute','relative','width','factor','coverage']	'absolute'
target	float|None	None
confidence	float	0.95
assumed_sd	float|None	None
assumed_mean	float|None	None
assumed_proportion	float|None	None
content	float|None	None
sided	Literal['one_lower','one_upper','two_sided']	'two_sided'
min_n	int	2
max_n	int	100000

min_n
Ge(ge=2); Le(le=1000000)

max_n
Ge(ge=2); Le(le=1000000)

4.1.396 PrecisionPlanningResult

`tabelz.analysis.precision_planning.PrecisionPlanningResult`

Immutable requested-versus-achieved precision plan.

Field	Type	Default
mode	Literal['mean','proportion','normal_tolerance','distribution_free_tolerance']	required
criterion	Literal['absolute','relative','width','factor','coverage']	required
requested_target	float	required
achieved_precision	float	required
achieved_criterion	bool	required
requested_confidence	float	required
achieved_confidence	float	required

Field	Type	Default
content	float|None	required
n	int	required
previous_n	int|None	required
previous_achieved_precision	float|None	required
previous_achieved_confidence	float|None	required
min_n	int	required
max_n	int	required
assumed_sd	float|None	required
assumed_mean	float|None	required
assumed_proportion	float|None	required
sided	Literal['one_lower','one_upper','two_sided']	required
method_identity	str	required
interval_convention	str	required
implementation_version	Literal['1.0.0']	'1.0.0'
warnings	tuple[str,...]	()
table_rows	tuple[tuple[object,...],...]	required
request_fingerprint	str	required

n
Ge(ge=2)

min_n
Ge(ge=2)

max_n
Ge(ge=2)

request_fingerprint
MinLen(min_length=64); MaxLen(max_length=64)

4.1.397 LevelMap

`tabelz.analysis.predictive_validation_contracts.LevelMap`

One categorical predictor's fitted training levels and reference.

Field	Type	Default
column	str	required
reference	str	required
indicator_levels	tuple[str,...]	required

indicator_levels
MinLen(min_length=1)

4.1.398 MetricSet

`tabelz.analysis.predictive_validation_contracts.MetricSet`

Point metrics for one row set; members are None when undefined.

Field	Type	Default
`row_count`	`int`	`required`
`rmse`	`float|None`	`None`
`mae`	`float|None`	`None`
`r_squared`	`float|None`	`None`

row_count
Ge(ge=0)

4.1.399 ScoringRecipe

`tabelz.analysis.predictive_validation_contracts.ScoringRecipe`

The retained training-fitted recipe that reproduces predictions.

Field	Type	Default
`terms`	`tuple[str,...]`	`required`
`numeric_fill_values`	`tuple[float|None,...]`	`required`
`level_maps`	`tuple[LevelMap,...]`	`required`
`scale_means`	`tuple[float,...]|None`	`required`
`scale_stds`	`tuple[float,...]|None`	`required`
`imputation`	`Literal['drop','mean','knn']`	`required`
`intercept`	`float`	`required`
`coefficients`	`tuple[float,...]`	`required`

4.1.400 SplitAssignment

`tabelz.analysis.predictive_validation_contracts.SplitAssignment`

Immutable, reusable training/test partition by stable row identity.

Field	Type	Default
`train_row_ids`	`tuple[str,...]`	`required`
`test_row_ids`	`tuple[str,...]`	`required`
`test_fraction`	`float`	`required`
`seed`	`int`	`required`

train_row_ids
MinLen(min_length=1)
test_row_ids
MinLen(min_length=1)

test_fraction
Gt(gt=0.0); Lt(lt=1.0)
seed
Ge(ge=0); Le(le=2147483647)

4.1.401 ValidatedRegressionConfig

`tabelz.analysis.predictive_validation_contracts.ValidatedRegressionConfig`

Frozen, strict validated-regression request.

Field	Type	Default
`response`	`str`	`required`
`numeric_predictors`	`tuple[str,...]`	`()`
`categorical_predictors`	`tuple[str,...]`	`()`
`algorithm`	`Literal['ols','ridge','lasso']`	`'ols'`
`imputation`	`Literal['drop','mean','knn']`	`'drop'`
`test_fraction`	`float`	`0.25`
`seed`	`int`	`0`
`cv_folds`	`int`	`5`
`max_rows`	`int`	`1000000`
`max_cells`	`int`	`20000000`

test_fraction
Ge(ge=0.05); Le(le=0.5)
seed
Ge(ge=0); Le(le=2147483647)
cv_folds
Ge(ge=2); Le(le=10)
max_rows
Ge(ge=1); Le(le=1000000)
max_cells
Ge(ge=1); Le(le=20000000)

4.1.402 ValidatedRegressionResult

`tabelz.analysis.predictive_validation_contracts.ValidatedRegressionResult`

Frozen, strict validated-regression outcome for transport and replay.

Field	Type	Default
`response`	`str`	`required`
`numeric_predictors`	`tuple[str,...]`	`required`
`categorical_predictors`	`tuple[str,...]`	`required`
`algorithm`	`Literal['ols','ridge','lasso']`	`required`

Field	Type	Default
imputation	Literal['drop','mean','knn']	required
test_fraction	float	required
seed	int	required
cv_folds	int	required
split	SplitAssignment	required
fold_validation_row_ids	tuple[tuple[str,...],...]	required
terms	tuple[str,...]	required
recipe	ScoringRecipe	required
training_metrics	MetricSet	required
test_metrics	MetricSet	required
cv_rmse	float|None	None
cv_curve_alphas	tuple[float,...]	()
cv_curve_rmse	tuple[float,...]	()
selected_alpha	float|None	None
source_count	int	required
excluded_count	int	required
missing_response_count	int	required
incomplete_categorical_count	int	required
incomplete_numeric_count	int	required
training_prediction_points	tuple[tuple[float,float],...]	required
test_prediction_points	tuple[tuple[float,float],...]	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_response_count
Ge(ge=0)
incomplete_categorical_count
Ge(ge=0)
incomplete_numeric_count
Ge(ge=0)

4.1.403 PcaConfig

`tabelz.analysis.principal_components.PcaConfig`

Frozen, strict PCA request: columns, scaling, and bounded-work limits.

Field	Type	Default
columns	tuple[str,...]	required
scaling	Literal['correlation','covariance']	'correlation'
n_components	int|None	None
max_rows	int	1000000
max_cells	int	20000000
max_scatter_points	int	5000

n_components
Ge(ge=1)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

max_scatter_points
Ge(ge=1); Le(le=5000)

4.1.404 PcaResult

tabelz.analysis.principal_components.PcaResult

Frozen, strict PCA outcome for transport and rendering.

Field	Type	Default
columns	tuple[str,...]	required
scaling	Literal['correlation','covariance']	required
component_count	int	required
means	tuple[float,...]	required
scales	tuple[float,...]	required
eigenvalues	tuple[float,...]	required
explained_variance_ratio	tuple[float,...]	required
cumulative_explained_variance	tuple[float,...]	required
loadings	tuple[tuple[float,...],...]	required
scores	tuple[tuple[float,...],...]	required
row_ids	tuple[str,...]	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
rank	int	required
warnings	tuple[str,...]	()

component_count
Ge(ge=1)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
rank
Ge(ge=0)

4.1.405 BatchTrajectory

`tabelz.analysis.product_stability.BatchTrajectory`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`batch_id`	`str`	`required`
`points`	`tuple[TrajectoryPoint,...]`	`required`

4.1.406 PoolabilityTestResult

`tabelz.analysis.product_stability.PoolabilityTestResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`f_stat`	`float`	`required`
`df_num`	`int`	`required`
`df_den`	`int`	`required`
`p_value`	`float`	`required`

4.1.407 ProductStabilityBatchResult

`tabelz.analysis.product_stability.ProductStabilityBatchResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`batch_id`	`str`	`required`
`intercept`	`float|None`	`None`
`slope`	`float|None`	`None`
`intercept_se`	`float|None`	`None`
`slope_se`	`float|None`	`None`
`shelf_life`	`float|None`	`None`

Field	Type	Default
shelf_life_status	str	required

4.1.408 ProductStabilityCoefficient

tabelz.analysis.product_stability.ProductStabilityCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
estimate	float	required
std_error	float	required
t_value	float	required
p_value	float	required

4.1.409 ProductStabilityConfig

tabelz.analysis.product_stability.ProductStabilityConfig

Replayable product stability request configuration.

Field	Type	Default
time_column	str	required
response_column	str	required
batch_column	str|None	None
model_type	Literal['pooled','separate_intercepts_pooled_slope','separate_intercepts_separate_slopes']	'pooled'
confidence_level	float	0.95
lsl	float|None	None
usl	float|None	None
max_rows	int	1000000
max_cells	int	20000000

confidence_level
Gt(gt=0.0); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=2); Le(le=20000000)

4.1.410 ProductStabilityPoolabilityTests

tabelz.analysis.product_stability.ProductStabilityPoolabilityTests

Common strict, immutable model baseline for public contracts.

Field	Type	Default
equal_slopes	PoolabilityTestResult|None	None
equal_intercepts	PoolabilityTestResult|None	None

4.1.411 ProductStabilityPrediction

tabelz.analysis.product_stability.ProductStabilityPrediction

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
time	float	required
observed	float|None	required
batch_id	str|None	required
fitted	float	required
lower_bound	float	required
upper_bound	float	required

4.1.412 ProductStabilityResult

tabelz.analysis.product_stability.ProductStabilityResult

Complete product shelf-life and stability studies result payload.

Field	Type	Default
config	ProductStabilityConfig	required
n	int	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
overall_shelf_life	float|None	None
overall_shelf_life_status	str	required
poolability_tests	ProductStabilityPoolabilityTests|None	None
batch_results	tuple[ProductStabilityBatchResult,...]	required
predictions	tuple[ProductStabilityPrediction,...]	required
trajectories	tuple[BatchTrajectory,...]	required
coefficients	tuple[ProductStabilityCoefficient,...]	required
warnings	tuple[str,...]	()

n
Ge(ge=1)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)

4.1.413 TrajectoryPoint

`tabelz.analysis.product_stability.TrajectoryPoint`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`time`	`float`	`required`
`fitted`	`float`	`required`
`lower_bound`	`float`	`required`
`upper_bound`	`float`	`required`

4.1.414 AdHocLineage

`tabelz.analysis.profiles.AdHocLineage`

Lineage for runs with no base profile.

Field	Type	Default
`lineage_kind`	`Literal['ad_hoc']`	`'ad_hoc'`

4.1.415 BooleanValue

`tabelz.analysis.profiles.BooleanValue`

Tagged value preserving the JSON boolean type.

Field	Type	Default
`value_type`	`Literal['boolean']`	`'boolean'`
`value`	`bool`	`required`

4.1.416 CustomRunProfile

`tabelz.analysis.profiles.CustomRunProfile`

Run profile carrying only a complete effective settings snapshot.

Field	Type	Default
profile_kind	Literal['custom']	'custom'
settings	tuple[ProfileSetting, ...]	required

4.1.417 FloatValue

tabelz.analysis.profiles.FloatValue

Tagged value preserving the JSON float type; values must be finite.

Field	Type	Default
value_type	Literal['float']	'float'
value	float	required

4.1.418 IntegerValue

tabelz.analysis.profiles.IntegerValue

Tagged value preserving the JSON integer type.

Field	Type	Default
value_type	Literal['integer']	'integer'
value	int	required

4.1.419 ModifiedLineage

tabelz.analysis.profiles.ModifiedLineage

Lineage naming the exact base and the settings whose values differ.

Field	Type	Default
lineage_kind	Literal['modified']	'modified'
base_name	str	required
base_profile_version	str	required
diverged_settings	tuple[ProfileSetting, ...]	required

base_name
MinLen(min_length=1)
base_profile_version
MinLen(min_length=1)

4.1.420 NullValue

tabelz.analysis.profiles.NullValue

Tagged value representing JSON null.

Field	Type	Default
value_type	Literal['null']	'null'
value	None	None

4.1.421 ProfileDefinition

tabelz.analysis.profiles.ProfileDefinition

Locked definition of one profile version and its complete settings.

Field	Type	Default
profile_schema_version	Literal['1.0.0']	'1.0.0'
name	str	required
profile_version	str	required
settings	tuple[ProfileSetting,...]	required
definition_fingerprint	str	required

name
MinLen(min_length=1)
profile_version
MinLen(min_length=1)

4.1.422 ProfileSetting

tabelz.analysis.profiles.ProfileSetting

One machine setting key paired with a strict tagged value.

Field	Type	Default
key	str	required
value	IntegerValue|FloatValue|NullValue|StringValue|BooleanValue	required

key
MinLen(min_length=1)

4.1.423 RegisteredLineage

tabelz.analysis.profiles.RegisteredLineage

Lineage naming the exact base profile with no divergences.

Field	Type	Default
lineage_kind	Literal['registered']	'registered'

Field	Type	Default
base_name	str	required
base_profile_version	str	required

base_name
MinLen(min_length=1)

base_profile_version
MinLen(min_length=1)

4.1.424 RegisteredRunProfile

tabelz.analysis.profiles.RegisteredRunProfile

Snapshot of a registered definition carried by a run.

Field	Type	Default
profile_kind	Literal['registered']	'registered'
name	str	required
profile_version	str	required
definition_fingerprint	str	required
settings	tuple[ProfileSetting, ...]	required

name
MinLen(min_length=1)

profile_version
MinLen(min_length=1)

4.1.425 StringValue

tabelz.analysis.profiles.StringValue

Tagged value preserving opaque strings, including date-like text.

Field	Type	Default
value_type	Literal['string']	'string'
value	str	required

4.1.426 QuantileCoefficient

tabelz.analysis.quantile_regression_contracts.QuantileCoefficient

One raw-scale coefficient with its interval and near-optimal sensitivity range.

Field	Type	Default
term	str	required
estimate	float	required
lower	float|None	None
upper	float|None	None
range_low	float|None	None
range_high	float|None	None
ambiguous	bool|None	False

4.1.427 QuantileEvaluation

tabelz.analysis.quantile_regression_contracts.QuantileEvaluation

Held-out check loss and the share of outcomes at or below the prediction.

Field	Type	Default
quantile	float	required
test_rows	int	required
mean_check_loss	float	required
at_or_below	int	required
ties	int	required
fraction_at_or_bel ow	float	required

test_rows
Ge(ge=1)

at_or_below
Ge(ge=0)

ties
Ge(ge=0)

4.1.428 QuantileFit

tabelz.analysis.quantile_regression_contracts.QuantileFit

Common strict, immutable model baseline for public contracts.

Field	Type	Default
quantile	float	required
coefficients	tuple[QuantileCoefficient,...]	required
check_loss	float	required
solver_status	str	required
primal_residual	float	required
objective_gap	float	0.0
dual_infeasibility	float	0.0

Field	Type	Default
duality_gap	float	0.0
probe_excess	float	0.0
sensitivity_note	str	''
minimizing_interval	tuple[float,float]|None	None
interval_status	Literal['available','tail_withheld', 'bootstrap_failed']	required
interval_note	str	''
resamples	int	required
failed_resamples	int	required
failure_examples	tuple[str,...]	()
tail_information	float	required

resamples
Ge(ge=0)

failed_resamples
Ge(ge=0)

4.1.429 QuantileGrid

tabelz.analysis.quantile_regression_contracts.QuantileGrid

Predicted quantile lines over one numeric predictor, other terms held fixed.

Field	Type	Default
predictor	str	required
held_fixed	str	required
x	tuple[float,...]	required
predictions	tuple[tuple[float,...],...]	required
crossing_points	int	required

crossing_points
Ge(ge=0)

4.1.430 QuantileRegressionConfig

tabelz.analysis.quantile_regression_contracts.QuantileRegressionConfig

Roles, quantiles, interval level, seed and the recorded bootstrap count.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	()
categorical_predictors	tuple[str,...]	()

Field	Type	Default
quantiles	tuple[float,...]	(0.5,)
confidence_level	float	0.95
dependence	Literal['independent','clustered','temporal']	'independent'
seed	int	0
held_out_evaluation	bool	False
bootstrap_resamples	int	999

confidence_level
Ge(ge=0.8); Le(le=0.99)
seed
Ge(ge=0); Le(le=2147483647)

4.1.431 QuantileRegressionResult

tabelz.analysis.quantile_regression_contracts.QuantileRegressionResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	required
categorical_predictors	tuple[str,...]	required
quantiles	tuple[float,...]	required
confidence_level	float	required
seed	int	required
bootstrap_resamples	int	required
terms	tuple[str,...]	required
fit_rows	int	required
source_count	int	required
excluded_count	int	required
incomplete_count	int	required
split	SplitAssignment|None	None
fits	tuple[QuantileFit,...]	required
evaluation	tuple[QuantileEvaluation,...]	()
crossing_rows	int	required
crossing_examples	tuple[str,...]	()
grid	QuantileGrid|None	None
points	tuple[tuple[float,float],...]	()
artifact	QuantileScoringArtifact	required
warnings	tuple[str,...]	()

fit_rows
Ge(ge=1)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
incomplete_count
Ge(ge=0)
crossing_rows
Ge(ge=0)

4.1.432 QuantileScoringArtifact

`tabelz.analysis.quantile_regression_contracts.QuantileScoringArtifact`

The fitting-space representation that reproduces conditional-quantile scores.

Field	Type	Default
`numeric_predictors`	`tuple[str,...]`	`required`
`level_maps`	`tuple[LevelMap,...]`	`required`
`terms`	`tuple[str,...]`	`required`
`quantiles`	`tuple[float,...]`	`required`
`intercepts`	`tuple[float,...]`	`required`
`coefficients`	`tuple[tuple[float,...],...]`	`required`
`centers`	`tuple[float,...]`	`required`
`offsets`	`tuple[float,...]`	`required`
`scales`	`tuple[float,...]`	`required`
`response_center`	`float`	`required`
`response_scale`	`float`	`required`
`fitted_coefficients`	`tuple[tuple[float,...],...]`	`required`

quantiles
MinLen(min_length=1); MaxLen(max_length=5)
response_scale
Ge(ge=0.0)

4.1.433 AlphaResult

`tabelz.analysis.rater_agreement.AlphaResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`metric`	`str`	`required`
`alpha`	`float|None`	`required`
`pairable_values`	`int`	`required`

Field	Type	Default
units	int	required
lower	float|None	required
upper	float|None	required
replicates	int	required
degenerate_replicates	int	required
seed	int	required
status	str	''

4.1.434 CategoryKappa

tabelz.analysis.rater_agreement.CategoryKappa

Common strict, immutable model baseline for public contracts.

Field	Type	Default
category	str	required
kappa	float|None	required
null_standard_error	float	required
z_value	float|None	required
p_value	float|None	required

4.1.435 KappaResult

tabelz.analysis.rater_agreement.KappaResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
units	int	required
kappa	float|None	required
observed_agreement	float	required
expected_agreement	float	required
standard_error	float|None	required
lower	float|None	required
upper	float|None	required
null_standard_error	float|None	required
z_value	float|None	required
p_value	float|None	required
bootstrap_lower	float|None	None
bootstrap_upper	float|None	None

Field	Type	Default
status	str	''

4.1.436 RaterAgreementConfig

tabelz.analysis.rater_agreement.RaterAgreementConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
raters	tuple[str,...]	required
categories	tuple[str,...]	()
weights	Literal['none','linear','quadratic']	'none'
alpha_metric	Literal['nominal','ordinal','interval']	'nominal'
confidence_level	float	0.95
replicates	int	5000
seed	int	20260928
kappa_bootstrap	bool	False

raters
MinLen(min_length=2); MaxLen(max_length=50)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
replicates
Ge(ge=200); Le(le=100000)
seed
Ge(ge=0)

4.1.437 RaterAgreementResult

tabelz.analysis.rater_agreement.RaterAgreementResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	RaterAgreementConfig	required
units	int	required
excluded_count	int	required
categories	tuple[str,...]	required
percent_agreement	float|None	required
complete_units	int	required
cohen	KappaResult|None	None
table	tuple[tuple[int,...],...]	()
fleiss	KappaResult|None	None

Field	Type	Default
`category_kappas`	`tuple[CategoryKappa, ...]`	`()`
`krippendorff`	`AlphaResult`	`required`
`notes`	`tuple[str, ...]`	`()`

4.1.438 BlockCoefficient

`tabelz.analysis.regression_blocks_contracts.BlockCoefficient`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`term`	`str`	`required`
`estimate`	`float`	`required`
`standard_error`	`float|None`	`required`
`standardized`	`float|None`	`required`
`t_value`	`float|None`	`required`
`p_value`	`float|None`	`required`
`lower`	`float|None`	`required`
`upper`	`float|None`	`required`
`zero_order`	`float|None`	`None`
`partial`	`float|None`	`None`
`part`	`float|None`	`None`
`tolerance`	`float|None`	`None`
`vif`	`float|None`	`None`

4.1.439 BlockModel

`tabelz.analysis.regression_blocks_contracts.BlockModel`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`block`	`int`	`required`
`predictors`	`tuple[str, ...]`	`required`
`entered`	`tuple[str, ...]`	`required`
`removed`	`tuple[str, ...]`	`()`
`r`	`float`	`required`
`r_squared`	`float`	`required`
`adjusted_r_squared`	`float`	`required`
`standard_error`	`float|None`	`required`
`r_squared_change`	`float`	`required`
`f_change`	`float|None`	`required`

Field	Type	Default
df1	int	required
df2	int	required
p_change	float|None	required
durbin_watson	float|None	required
regression_ss	float	required
residual_ss	float	required
f_value	float|None	required
p_value	float|None	required
coefficients	tuple[BlockCoefficient,...]	required
collinearity	tuple[CollinearityDimension,...]	required
collinearity_terms	tuple[str,...]	required
selection	tuple[SelectionMove,...]	()

4.1.440 CasewiseRow

tabelz.analysis.regression_blocks_contracts.CasewiseRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
observed	float	required
predicted	float	required
residual	float	required
standardized_residual	float|None	required
studentized_residual	float|None	required
deleted_studentized_residual	float|None	required
leverage	float|None	required
cooks_distance	float|None	required
dfbetas	tuple[float|None,...]	required

4.1.441 CollinearityDimension

tabelz.analysis.regression_blocks_contracts.CollinearityDimension

Common strict, immutable model baseline for public contracts.

Field	Type	Default
dimension	int	required
eigenvalue	float	required

Field	Type	Default
condition_index	float|None	required
proportions	tuple[float|None,...]	required

4.1.442 RegressionBlocksConfig

tabelz.analysis.regression_blocks_contracts.RegressionBlocksConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
blocks	tuple[tuple[str,...],...]	required
method	Literal['enter','forward','backward', 'stepwise']	'enter'
criterion	Literal['aic','bic']	'aic'
confidence_level	float	0.95
casewise	Literal['outliers','all','none']	'outliers'
casewise_threshold	float	3.0
frequency_weighted	bool	False

blocks
MinLen(min_length=1); MaxLen(max_length=10)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
casewise_threshold
Gt(gt=0.0); Le(le=100.0)

4.1.443 RegressionBlocksResult

tabelz.analysis.regression_blocks_contracts.RegressionBlocksResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	RegressionBlocksConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
models	tuple[BlockModel,...]	required
casewise_terms	tuple[str,...]	()
casewise	tuple[CasewiseRow,...]	()
casewise_truncated	bool	False
design_terms	tuple[str,...]	()
design_collinearity	tuple[CollinearityDimension,...]	()

Field	Type	Default
notes	tuple[str,...]	()
warnings	tuple[str,...]	()

4.1.444 SelectionMove

tabelz.analysis.regression_blocks_contracts.SelectionMove

Common strict, immutable model baseline for public contracts.

Field	Type	Default
action	Literal['start','add','drop']	required
predictor	str	required
criterion	float	required

4.1.445 ForestCandidate

tabelz.analysis.regression_forest_contracts.ForestCandidate

One grid point. “nominal_candidate_features“ is scikit-learn’s per-split draw, max(1, floor(max_features x encoded columns)); it may inspect more columns when none of the drawn ones gives a valid split, so it is not an unconditional bound.

Field	Type	Default
settings	ForestSettings	required
nominal_candidate_features	int	required
fold_rmse	tuple[float|None,...]	required
cv_rmse	float|None	None
status	Literal['ok','failed']	required
failure	str	''

nominal_candidate_features
Ge(ge=1)

4.1.446 ForestEncoding

tabelz.analysis.regression_forest_contracts.ForestEncoding

Imputation, numeric scaling and category dictionaries fitted on the model’s rows.

Field	Type	Default
numeric_predictors	tuple[str,...]	required
numeric_medians	tuple[float,...]	required
numeric_scales	tuple[float,...]	required

Field	Type	Default
categorical_predictors	tuple[str,...]	required
categorical_levels	tuple[tuple[str|None,...],...]	required

4.1.447 ForestImportance

tabelz.analysis.regression_forest_contracts.ForestImportance

Common strict, immutable model baseline for public contracts.

Field	Type	Default
predictor	str	required
mean_rmse_increase	float	required
sd_rmse_increase	float	required
permutations	int	required

permutations
Ge(ge=2)

4.1.448 ForestModelRef

tabelz.analysis.regression_forest_contracts.ForestModelRef

A fitted forest's tree state, named by the SHA-256 of its compressed container.

Field	Type	Default
sha256	str	required
compressed_bytes	int	required
trees	int	required
nodes	int	required
features	int	required
data	str|None	None

compressed_bytes
Ge(ge=1)

trees
Ge(ge=50); Le(le=500)

nodes
Ge(ge=1)

features
Ge(ge=1)

4.1.449 ForestSettings

tabelz.analysis.regression_forest_contracts.ForestSettings

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`min_samples_leaf`	`int`	`required`
`max_features`	`float`	`required`
`max_depth`	`int|None`	`None`

min_samples_leaf
Ge(ge=1)

max_features
Gt(gt=0.0); Le(le=1.0)

max_depth
Ge(ge=1)

4.1.450 RegressionForestArtifact

`tabelz.analysis.regression_forest_contracts.RegressionForestArtifact`

One fitted forest and the frozen encoding its rows pass through.

Field	Type	Default
`fit`	`Literal['training','deployment']`	`required`
`response`	`str`	`required`
`encoding`	`ForestEncoding`	`required`
`settings`	`ForestSettings`	`required`
`trees`	`int`	`required`
`chunk_seeds`	`tuple[int,...]`	`required`
`fit_row_count`	`int`	`required`
`scored_row_ids`	`tuple[str,...]`	`required`
`model`	`ForestModelRef`	`required`
`sklearn_version`	`str`	`required`

trees
Ge(ge=50); Le(le=500)

chunk_seeds
MinLen(min_length=1)

fit_row_count
Ge(ge=2)

scored_row_ids
MinLen(min_length=1)

4.1.451 RegressionForestConfig

`tabelz.analysis.regression_forest_contracts.RegressionForestConfig`

One continuous response, numeric and categorical predictors and a seed.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	()
categorical_predictors	tuple[str,...]	()
dependence	Literal['independent','clustered','temporal']	'independent'
seed	int	0
trees	int	500

seed
Ge(ge=0); Le(le=2147483647)

4.1.452 RegressionForestResult

tabelz.analysis.regression_forest_contracts.RegressionForestResult

The saved study: split, grid, selection, first test evaluation and artifacts.

Field	Type	Default
response	str	required
numeric_predictors	tuple[str,...]	required
categorical_predictors	tuple[str,...]	required
seed	int	required
trees	int	required
encoded_feature_count	int	required
split	SplitAssignment	required
fold_validation_row_ids	tuple[tuple[str,...],...]	required
candidates	tuple[ForestCandidate,...]	required
selected_index	int	required
baseline_fold_rmse	tuple[float,...]	required
baseline_cv_rmse	float	required
forest_improves_baseline	bool	required
validation_predictions	tuple[float,...]	required
test_predictions	tuple[float,...]	required
test_metrics	MetricSet	required
baseline_test_metrics	MetricSet	required
importance_base_rmse	float	required

Field	Type	Default
importance	tuple[ForestImportance,...]	required
validation_points	tuple[tuple[float,float],...]	required
test_points	tuple[tuple[float,float],...]	required
artifact	RegressionForestArtifact	required
deployment_artifact	RegressionForestArtifact	required
source_count	int	required
excluded_count	int	required
missing_response_count	int	required
warnings	tuple[str,...]	()

encoded_feature_count
Ge(ge=1)
selected_index
Ge(ge=0)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_response_count
Ge(ge=0)

4.1.453 RegressionTreeArtifact

tabelz.analysis.regression_tree.RegressionTreeArtifact

Safe, immutable numeric representation of a fitted regression tree.

Field	Type	Default
feature_order	tuple[str,...]	required
children_left	tuple[int,...]	required
children_right	tuple[int,...]	required
feature	tuple[int,...]	required
threshold	tuple[float,...]	required
value	tuple[float,...]	required
n_node_samples	tuple[int,...]	required

feature_order
MinLen(min_length=1); MaxLen(max_length=64)
children_left
MinLen(min_length=1); MaxLen(max_length=511)
children_right
MinLen(min_length=1); MaxLen(max_length=511)

feature
MinLen(min_length=1); MaxLen(max_length=511)
threshold
MinLen(min_length=1); MaxLen(max_length=511)
value
MinLen(min_length=1); MaxLen(max_length=511)
n_node_samples
MinLen(min_length=1); MaxLen(max_length=511)

4.1.454 RegressionTreeCandidate

`tabelz.analysis.regression_tree.RegressionTreeCandidate`

One training-only CV candidate in deterministic grid order.

Field	Type	Default
`max_depth`	`int`	`required`
`min_samples_leaf`	`int`	`required`
`cv_rmse`	`float`	`required`

max_depth
Ge(ge=1); Le(le=8)
min_samples_leaf
Ge(ge=1); Le(le=100000)
cv_rmse
Ge(ge=0)

4.1.455 RegressionTreeConfig

`tabelz.analysis.regression_tree.RegressionTreeConfig`

Strict, immutable F15 request.

Field	Type	Default
`response`	`str`	`required`
`numeric_predictors`	`tuple[str,...]`	`required`
`test_fraction`	`float`	`0.25`
`seed`	`int`	`0`
`cv_folds`	`int`	`5`
`max_depth`	`int`	`8`
`min_samples_leaf`	`int`	`1`
`depth_grid`	`tuple[int,...]`	`()`
`min_leaf_grid`	`tuple[int,...]`	`()`
`max_rows`	`int`	`1000000`
`max_cells`	`int`	`20000000`

numeric_predictors
MinLen(min_length=1); MaxLen(max_length=64)

test_fraction
Ge(ge=0.05); Le(le=0.5)

seed
Ge(ge=0); Le(le=2147483647)

cv_folds
Ge(ge=2); Le(le=10)

max_depth
Ge(ge=1); Le(le=8)

min_samples_leaf
Ge(ge=1); Le(le=100000)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=20000000)

4.1.456 RegressionTreeMetricSet

`tabelz.analysis.regression_tree.RegressionTreeMetricSet`

RMSE/MAE and optional R-squared for one fixed row set.

Field	Type	Default
`row_count`	`int`	`required`
`rmse`	`float|None`	`None`
`mae`	`float|None`	`None`
`r_squared`	`float|None`	`None`

row_count
Ge(ge=0)

4.1.457 RegressionTreeResult

`tabelz.analysis.regression_tree.RegressionTreeResult`

Replayable F15 result and bounded diagnostics.

Field	Type	Default
`response`	`str`	`required`
`numeric_predictors`	`tuple[str,...]`	`required`
`test_fraction`	`float`	`required`
`seed`	`int`	`required`
`cv_folds`	`int`	`required`
`max_depth`	`int`	`required`
`min_samples_leaf`	`int`	`required`

Field	Type	Default
depth_grid	tuple[int,...]	required
min_leaf_grid	tuple[int,...]	required
split	SplitAssignment	required
fold_validation_row_ids	tuple[tuple[str,...],...]	required
selected_max_depth	int	required
selected_min_samples_leaf	int	required
selected_cv_rmse	float	required
candidates	tuple[RegressionTreeCandidate,...]	required
artifact	RegressionTreeArtifact	required
training_metrics	RegressionTreeMetricSet	required
test_metrics	RegressionTreeMetricSet	required
test_baseline_metrics	RegressionTreeMetricSet	required
training_mean	float	required
feature_importances	tuple[float,...]	required
split_rows	tuple[tuple[int,str,float,int,int,int],...]	required
leaf_rows	tuple[tuple[int,int,str,float,int],...]	required
training_prediction_points	tuple[tuple[float,float],...]	required
test_prediction_points	tuple[tuple[float,float],...]	required
source_count	int	required
excluded_count	int	required
missing_response_count	int	required
incomplete_numeric_count	int	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)

excluded_count
Ge(ge=0)

missing_response_count
Ge(ge=0)

incomplete_numeric_count
Ge(ge=0)

4.1.458 IccRow

`tabelz.analysis.reliability_scale_contracts.IccRow`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`label`	`str`	required
`shrout_fleiss`	`str`	required
`mcgraw_wong`	`str`	required
`estimate`	`float|None`	required
`lower`	`float|None`	required
`upper`	`float|None`	required
`interval`	`str`	required
`f_value`	`float|None`	required
`df1`	`float`	required
`df2`	`float`	required
`p_value`	`float|None`	required
`status`	`str`	`''`

4.1.459 ItemTotal

`tabelz.analysis.reliability_scale_contracts.ItemTotal`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`item`	`str`	required
`reversed`	`bool`	required
`mean`	`float`	required
`sd`	`float`	required
`scale_mean_if_deleted`	`float`	required
`scale_variance_if_deleted`	`float`	required
`corrected_item_total`	`float|None`	required
`squared_multiple_correlation`	`float|None`	required
`alpha_if_deleted`	`float|None`	required

4.1.460 ModelFit

`tabelz.analysis.reliability_scale_contracts.ModelFit`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
model	Literal['parallel','strictlyparallel']	required
chi_square	float|None	required
df	int	required
p_value	float|None	required
log_det_unconstrained	float|None	required
log_det_constrained	float|None	required
common_variance	float	required
true_variance	float	required
error_variance	float	required
common_correlation	float|None	required
scale_reliability	float|None	required
note	str	''

4.1.461 ScaleReliabilityConfig

tabelz.analysis.reliability_scale_contracts.ScaleReliabilityConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
items	tuple[str,...]	required
reverse_items	tuple[str,...]	()
scale_low	float|None	None
scale_high	float|None	None
split	Literal['halves','odd_even','custom']	'halves'
first_half	tuple[str,...]	()
icc_model	Literal['none','one_way','two_way_random','two_way_mixed']	'two_way_mixed'
icc_type	Literal['consistency','agreement']	'consistency'
confidence_level	float	0.95

items
MinLen(min_length=2); MaxLen(max_length=64)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.462 ScaleReliabilityResult

tabelz.analysis.reliability_scale_contracts.ScaleReliabilityResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ScaleReliabilityConfig	required
n	int	required
k	int	required
missing_count	int	required
excluded_count	int	required
alpha	float|None	required
standardized_alpha	float|None	required
guttman	tuple[float|None,...]	required
split_half	SplitHalf	required
summaries	tuple[SummaryStatistic,...]	required
item_totals	tuple[ItemTotal,...]	required
models	tuple[ModelFit,...]	required
tests	tuple[TestRow,...]	required
anova	tuple[tuple[str,float,int,float|None],...]	required
icc	tuple[IccRow,...]	()
notes	tuple[str,...]	()

4.1.463 SplitHalf

tabelz.analysis.reliability_scale_contracts.SplitHalf

Common strict, immutable model baseline for public contracts.

Field	Type	Default
first_items	tuple[str,...]	required
second_items	tuple[str,...]	required
first_alpha	float|None	required
second_alpha	float|None	required
correlation	float|None	required
spearman_brown_equal	float|None	required
spearman_brown_unequal	float|None	required
guttman_split_half	float|None	required
note	str	''

4.1.464 SummaryStatistic

tabelz.analysis.reliability_scale_contracts.SummaryStatistic

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
mean	float	required
minimum	float	required
maximum	float	required
range	float	required
ratio	float|None	required
variance	float	required
count	int	required

4.1.465 TestRow

tabelz.analysis.reliability_scale_contracts.TestRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
statistic	float|None	required
df1	float|None	required
df2	float|None	None
p_value	float|None	required
status	str	''

4.1.466 ContrastEstimate

tabelz.analysis.repeated_covariance_contracts.ContrastEstimate

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
weights	dict[str,float]	required
estimate	float	required
std_error	float	required
df	float|None	required
t_value	float|None	required
p_value	float|None	required
lower	float|None	required
upper	float|None	required

4.1.467 ContrastSpec

tabelz.analysis.repeated_covariance_contracts.ContrastSpec

A named linear combination of fixed coefficients (term name -> weight).

Field	Type	Default
name	str	required
weights	dict[str,float]	required

name
MinLen(min_length=1); MaxLen(max_length=120)
weights
MinLen(min_length=1)

4.1.468 FittedRow

tabelz.analysis.repeated_covariance_contracts.FittedRow

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_id	str	required
subject	str	required
occasion	str	required
observed	float	required
fitted	float	required
lower	float|None	required
upper	float|None	required
residual	float	required
normalized_residual	float	required

4.1.469 FixedCoefficient

tabelz.analysis.repeated_covariance_contracts.FixedCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
column	str|None	required
reference_level	str|None	required
estimate	float	required
std_error	float	required
df	float|None	required
t_value	float|None	required
p_value	float|None	required
lower	float|None	required

Field	Type	Default
upper	float|None	required

4.1.470 ModelComparison

tabelz.analysis.repeated_covariance_contracts.ModelComparison

ML diagnostics of nested fixed models on identical rows and covariance.

Field	Type	Default
full_terms	tuple[str,...]	required
reduced_terms	tuple[str,...]	required
full_log_likelihood	float	required
reduced_log_likelihood	float	required
likelihood_ratio	float	required
df_difference	int	required
full_aic	float	required
reduced_aic	float	required
full_bic	float	required
reduced_bic	float	required
reduced_converged	bool	required

4.1.471 OccasionLevel

tabelz.analysis.repeated_covariance_contracts.OccasionLevel

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
position	float|None	required
observations	int	required

4.1.472 OccasionSummary

tabelz.analysis.repeated_covariance_contracts.OccasionSummary

Common strict, immutable model baseline for public contracts.

Field	Type	Default
occasion	str	required
observations	int	required
mean_residual	float	required
model_variance	float	required

Field	Type	Default
empirical_residual_variance	float|None	required

4.1.473 RepeatedCovarianceConfig

tabelz.analysis.repeated_covariance_contracts.RepeatedCovarianceConfig

A replayable repeated-measures covariance request.

Field	Type	Default
response_column	str	required
subject_column	str	required
occasion_column	str	required
fixed_numeric_columns	tuple[str,...]	()
fixed_categorical_columns	tuple[str,...]	()
residual_structure	Literal['independent','ar1','heterogeneous','unstructured']	'independent'
random_intercept	bool	False
estimation	Literal['reml','ml']	'reml'
occasion_spacing	float|None	None
confidence_level	float	0.95
contrasts	tuple[ContrastSpec,...]	()
reduced_fixed_columns	tuple[str,...]|None	None
max_rows	int	20000

response_column
MinLen(min_length=1)
subject_column
MinLen(min_length=1)
occasion_column
MinLen(min_length=1)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
max_rows
Ge(ge=3); Le(le=1000000)

4.1.474 RepeatedCovarianceResult

tabelz.analysis.repeated_covariance_contracts.RepeatedCovarianceResult

A repeated-measures covariance fit, its inference state and diagnostics.

Field	Type	Default
method_identity	Literal['models.repeated_covariance']	'models.repeated_covariance'
implementation_version	Literal['1.0.0']	'1.0.0'
config	RepeatedCovarianceConfig	required
status	Literal['ok','inference_unavailable', 'not_converged']	required
structure_label	str	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
analyzed_count	int	required
subject_count	int	required
occasions	tuple[OccasionLevel,...]	required
max_observations_per_subject	int	required
coefficients	tuple[FixedCoefficient,...]	required
contrasts	tuple[ContrastEstimate,...]	required
variance_parameters	tuple[VarianceParameter,...]	required
covariance_matrix	tuple[tuple[float,...],...]	required
correlation_matrix	tuple[tuple[float|None,...],...]	required
log_likelihood	float	required
criterion	Literal['reml','ml']	required
parameter_count	int	required
bic_sample_size	int	required
aic	float	required
bic	float	required
converged	bool	required
iterations	int	required
gradient_norm	float	required
optimizer_message	str	required
random_intercept_boundary	bool	required
boundary_objective	float|None	required
interior_objective	float|None	required
inference_available	bool	required
inference_unavailable_reason	str|None	required
inference_label	str	required
fitted	tuple[FittedRow,...]	required
fitted_truncated	bool	required

Field	Type	Default
occasion_summaries	tuple[OccasionSummary,...]	required
comparison	ModelComparison|None	required
warnings	tuple[str,...]	required
population_fingerprint	str	required

4.1.475 VarianceParameter

tabelz.analysis.repeated_covariance_contracts.VarianceParameter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
approximate_se	float|None	required
at_boundary	bool	required

4.1.476 BetweenCell

tabelz.analysis.repeated_measures_contracts.BetweenCell

Common strict, immutable model baseline for public contracts.

Field	Type	Default
levels	tuple[str,...]	required
n	int	required

4.1.477 BetweenTest

tabelz.analysis.repeated_measures_contracts.BetweenTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
effect	str	required
sum_of_squares	float	required
df	int	required
mean_square	float	required
f_value	float|None	required
p_value	float|None	required
partial_eta_squared	float|None	required

4.1.478 CellAssignment

`tabelz.analysis.repeated_measures_contracts.CellAssignment`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`levels`	`tuple[str,...]`	`required`
`column`	`str`	`required`
`mean`	`float`	`required`
`sd`	`float`	`required`

4.1.479 ContrastTest

`tabelz.analysis.repeated_measures_contracts.ContrastTest`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`factor`	`str`	`required`
`name`	`str`	`required`
`coefficients`	`tuple[float,...]`	`required`
`estimate`	`float`	`required`
`standard_error`	`float`	`required`
`df`	`int`	`required`
`t_value`	`float`	`required`
`p_value`	`float`	`required`
`p_adjusted`	`float|None`	`None`
`lower`	`float|None`	`None`
`upper`	`float|None`	`None`

4.1.480 CorrectedTest

`tabelz.analysis.repeated_measures_contracts.CorrectedTest`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`correction`	`Literal['Sphericityassumed','Greenhouse-Geisser','Huynh-Feldt','Lowerbound']`	`required`
`epsilon`	`float|None`	`None`
`df1`	`float|None`	`None`
`df2`	`float|None`	`None`
`p_value`	`float|None`	`None`

4.1.481 DesignSummary

`tabelz.analysis.repeated_measures_contracts.DesignSummary`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`subjects`	`int`	`required`
`used`	`int`	`required`
`incomplete`	`int`	`required`
`excluded`	`int`	`required`
`residual_df`	`int`	`required`
`cells`	`tuple[CellAssignment,...]`	`required`
`between_cells`	`tuple[BetweenCell,...]`	`()`
`covariate_means`	`tuple[tuple[str,float],...]`	`()`

4.1.482 InteractionContrast

`tabelz.analysis.repeated_measures_contracts.InteractionContrast`

A crossed scalar contrast ell′ B d; both factors′ coefficient vectors are kept.

Field	Type	Default
`kind`	`Literal['within×within','between×within']`	`required`
`first_factor`	`str`	`required`
`first_contrast`	`str`	`required`
`first_coefficients`	`tuple[float,...]`	`required`
`second_factor`	`str`	`required`
`second_contrast`	`str`	`required`
`second_coefficients`	`tuple[float,...]`	`required`
`estimate`	`float|None`	`required`
`standard_error`	`float|None`	`required`
`df`	`int`	`required`
`t_value`	`float|None`	`required`
`p_value`	`float|None`	`required`
`p_adjusted`	`float|None`	`None`
`interval`	`Literal['unadjusted','bonferroni']`	`'unadjusted'`
`lower`	`float|None`	`required`
`upper`	`float|None`	`required`
`unavailable_reason`	`str`	`''`

4.1.483 MarginalMean

`tabelz.analysis.repeated_measures_contracts.MarginalMean`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
within	tuple[str,...]	()
between	tuple[str,...]	()
mean	float	required
standard_error	float	required
lower	float	required
upper	float	required

4.1.484 MarginalTable

tabelz.analysis.repeated_measures_contracts.MarginalTable

Common strict, immutable model baseline for public contracts.

Field	Type	Default
title	str	required
within_factors	tuple[str,...]	()
between_factors	tuple[str,...]	()
rows	tuple[MarginalMean,...]	required

4.1.485 MultivariateTest

tabelz.analysis.repeated_measures_contracts.MultivariateTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
effect	str	required
statistic	Literal["Pillai'strace","Wilks'lambda",'Hotelling-Lawleytrace',"Roy'sroot"]	required
value	float|None	None
f_value	float|None	None
df1	float|None	None
df2	float|None	None
p_value	float|None	None
exact	bool	False
note	str	''

4.1.486 PairwiseFamily

tabelz.analysis.repeated_measures_contracts.PairwiseFamily

Common strict, immutable model baseline for public contracts.

Field	Type	Default
factor	str	required
kind	Literal['within','between']	required
result	ComparisonResult	required

4.1.487 ProfilePlot

tabelz.analysis.repeated_measures_contracts.ProfilePlot

Common strict, immutable model baseline for public contracts.

Field	Type	Default
title	str	required
x_factor	str	required
x_levels	tuple[str,...]	required
line_factor	str	''
series	tuple[ProfileSeries,...]	required

4.1.488 ProfileSeries

tabelz.analysis.repeated_measures_contracts.ProfileSeries

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
means	tuple[float,...]	required
lower	tuple[float,...]	required
upper	tuple[float,...]	required

4.1.489 RepeatedMeasuresConfig

tabelz.analysis.repeated_measures_contracts.RepeatedMeasuresConfig

Settings as entered: up to two within factors (name, ordered levels and optional level values that declare an ordered quantitative factor) and one column per cell.

Field	Type	Default
factor1_name	str	required
factor1_levels	tuple[str,...]	required
factor1_values	tuple[float,...]	()
factor2_name	str	''
factor2_levels	tuple[str,...]	()
factor2_values	tuple[float,...]	()
cells	tuple[str,...]	required

Field	Type	Default
between	tuple[str,...]	()
covariates	tuple[str,...]	()
covariate_centering	Literal['mean','none']	'mean'
contrast_family	Literal['auto','none','deviation','simple_first','simple_last','difference','helmert','repeated','polynomial']	'auto'
contrast_adjustment	Literal['none','bonferroni','holm']	'none'
emm_averaging	Literal['equal','observed']	'equal'
pairwise_procedure	Literal['bonferroni','sidak','holm','none']	'bonferroni'
alpha	float	0.05
simple_effects	bool	False
interaction_contrasts	bool	False
followup_adjustment	Literal['none','bonferroni','holm','sidak']	'none'

factor1_name
MinLen(min_length=1); MaxLen(max_length=64)
factor1_levels
MinLen(min_length=2); MaxLen(max_length=20)
factor2_name
MaxLen(max_length=64)
factor2_levels
MaxLen(max_length=20)
cells
MinLen(min_length=2); MaxLen(max_length=60)
between
MaxLen(max_length=3)
covariates
MaxLen(max_length=5)
alpha
Gt(gt=0.0); Lt(lt=0.5)

4.1.490 RepeatedMeasuresResult

`tabelz.analysis.repeated_measures_contracts.RepeatedMeasuresResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	RepeatedMeasuresConfig	required
design	DesignSummary	required
sphericity	tuple[Sphericity,...]	required

Field	Type	Default
within_tests	tuple[UnivariateTest,...]	required
multivariate_tests	tuple[MultivariateTest,...]	required
between_tests	tuple[BetweenTest,...]	required
between_error_ss	float	required
contrasts	tuple[ContrastTest,...]	()
marginal_means	tuple[MarginalTable,...]	()
pairwise	tuple[PairwiseFamily,...]	()
profiles	tuple[ProfilePlot,...]	()
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
simple_effects	tuple[SimpleEffectSet,...]	()
interaction_contrasts	tuple[InteractionContrast,...]	()

4.1.491 SimpleEffect

tabelz.analysis.repeated_measures_contracts.SimpleEffect

One conditioning stratum of a simple-effect set (ruling #111).

Field	Type	Default
effect	str	required
level	str	required
dimensions	int	required
hypothesis_rank	int	required
sphericity	Sphericity|None	None
univariate	UnivariateTest|None	None
univariate_p_adjusted	tuple[float|None,...]	()
multivariate	tuple[MultivariateTest,...]	()
multivariate_p_adjusted	tuple[float|None,...]	()
unavailable_reason	str	''

4.1.492 SimpleEffectSet

tabelz.analysis.repeated_measures_contracts.SimpleEffectSet

One tested factor at every level of one conditioning factor: one multiplicity family.

Field	Type	Default
tested	str	required
tested_kind	Literal['within','between']	required

Field	Type	Default
conditioning	str	required
conditioning_kind	Literal['within','between']	required
adjustment	Literal['none','bonferroni','holm','sidak']	required
family	str	required
effects	tuple[SimpleEffect,...]	required
note	str	''

4.1.493 Sphericity

tabelz.analysis.repeated_measures_contracts.Sphericity

Common strict, immutable model baseline for public contracts.

Field	Type	Default
effect	str	required
dimensions	int	required
mauchly_w	float|None	None
chi_square	float|None	None
df	int|None	None
p_value	float|None	None
greenhouse_geisser	float|None	None
huynh_feldt	float|None	None
huynh_feldt_applied	float|None	None
lower_bound	float	required
note	str	''

4.1.494 UnivariateTest

tabelz.analysis.repeated_measures_contracts.UnivariateTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
effect	str	required
sum_of_squares	float	required
df	int	required
mean_square	float	required
f_value	float|None	required
error_sum_of_squares	float	required
error_df	int	required
partial_eta_squared	float|None	required

Field	Type	Default
tests	tuple[CorrectedTest,...]	required

4.1.495 ColumnComparison

tabelz.analysis.report_custom_tables.ColumnComparison

Common strict, immutable model baseline for public contracts.

Field	Type	Default
layer	str	required
group	tuple[str,...]	required
subject	str	required
first	str	required
second	str	required
statistic	float|None	required
p_value	float|None	required
adjusted_p	float|None	required
family_size	int	required
note	str	''

4.1.496 CustomTableConfig

tabelz.analysis.report_custom_tables.CustomTableConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
metadata	dict[str,ColumnMetadata]	factory:dict
treat_user_missing_as_valid	Literal[True]	True
rows	tuple[str,...]	required
columns	tuple[str,...]	()
layer	str|None	None
summary	str|None	None
sets	dict[str,ResponseSet]	factory:dict
statistics	tuple[Literal['count','row_percent','column_percent','layer_percent','table_percent','mean','median','sd','min','max','sum','valid_n','responses'],...]	('count','column_percent')
totals	bool	True
independence	bool	False
compare	Literal['none','proportions','means']	'none'

Field	Type	Default
adjustment	Literal['holm','bonferroni','none']	'holm'
alpha	float	0.05
structural_zeros	tuple[StructuralZero,...]	()

rows
MinLen(min_length=1); MaxLen(max_length=3)

columns
MaxLen(max_length=3)

alpha
Gt(gt=0.0); Lt(lt=0.5)

4.1.497 CustomTableResult

tabelz.analysis.report_custom_tables.CustomTableResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CustomTableConfig	required
layers	tuple[str,...]	required
row_keys	tuple[tuple[str,...],...]	required
column_keys	tuple[tuple[str,...],...]	required
column_letters	tuple[str,...]	required
cells	tuple[TableCell,...]	required
independence	tuple[IndependenceTest,...]	()
comparisons	tuple[ColumnComparison,...]	()
used_rows	int	required
missing_rows	int	required
notes	tuple[str,...]	()

4.1.498 IndependenceTest

tabelz.analysis.report_custom_tables.IndependenceTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
layer	str	required
variable	str	required
chi_square	float|None	required
df	int	required
p_value	float|None	required
min_expected	float|None	required

Field	Type	Default
small_expected	int	required
cells	int	required
note	str	''

4.1.499 StructuralZero

tabelz.analysis.report_custom_tables.StructuralZero

A user-declared impossible combination of a row level and a column level (#146).

Field	Type	Default
row_variable	str	required
row_level	str	required
column_variable	str	required
column_level	str	required

4.1.500 TableCell

tabelz.analysis.report_custom_tables.TableCell

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row	tuple[str,...]	required
column	tuple[str,...]	required
statistic	str	required
value	float|None	required
flags	str	''

4.1.501 CaseSummariesConfig

tabelz.analysis.report_tables_contracts.CaseSummariesConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
metadata	dict[str,ColumnMetadata]	factory:dict
treat_user_missing_as_valid	Literal[True]	True
columns	tuple[str,...]	required
group	str|None	None
limit	int	100
statistics	bool	True

Field	Type	Default
frequency_weight	str|None	None

columns
MinLen(min_length=1); MaxLen(max_length=50)
limit
Ge(ge=1); Le(le=5000)

4.1.502 CaseSummariesResult

tabelz.analysis.report_tables_contracts.CaseSummariesResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CaseSummariesConfig	required
columns	tuple[str,...]	required
rows	tuple[tuple[str,...],...]	required
listed	int	required
total	int	required
statistics	tuple[GroupStatistic,...]	()
notes	tuple[str,...]	()

4.1.503 CodebookConfig

tabelz.analysis.report_tables_contracts.CodebookConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
metadata	dict[str,ColumnMetadata]	factory:dict
treat_user_missing_as_valid	Literal[True]	True
columns	tuple[str,...]	required

columns
MinLen(min_length=1); MaxLen(max_length=500)

4.1.504 CodebookEntry

tabelz.analysis.report_tables_contracts.CodebookEntry

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required

Field	Type	Default
label	str	required
storage	str	required
level	str	required
display	str	required
user_missing	str	required
values	tuple[ValueCount, ...]	()
summary	tuple[tuple[str, float|None], ...]	()
valid	float	required
user_missing_count	float	required
system_missing_count	float	required

4.1.505 CodebookResult

tabelz.analysis.report_tables_contracts.CodebookResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	CodebookConfig	required
rows	int	required
entries	tuple[CodebookEntry, ...]	required
notes	tuple[str, ...]	()

4.1.506 ExploreBox

tabelz.analysis.report_tables_contracts.ExploreBox

Common strict, immutable model baseline for public contracts.

Field	Type	Default
q1	float	required
median	float	required
q3	float	required
whisker_low	float	required
whisker_high	float	required
outliers	tuple[tuple[int|None, float], ...]	required
extremes	tuple[tuple[int|None, float], ...]	required

4.1.507 ExploreCell

tabelz.analysis.report_tables_contracts.ExploreCell

Common strict, immutable model baseline for public contracts.

Field	Type	Default
dependent	str	required
group	str	required
n	int	required
missing	int	required
descriptives	tuple[tuple[str,float|None,float|None],...]	required
mean_lower	float|None	required
mean_upper	float|None	required
m_estimates	tuple[ExploreMEstimate,...]	required
extremes	tuple[ExploreExtreme,...]	required
normality	tuple[ExploreNormality,...]	required
stem_leaf	StemLeafResult|None	required
box	ExploreBox|None	required
qq	tuple[tuple[float,float],...]	required

4.1.508 ExploreConfig

tabelz.analysis.report_tables_contracts.ExploreConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
metadata	dict[str,ColumnMetadata]	factory:dict
treat_user_missing_as_valid	Literal[True]	True
dependents	tuple[str,...]	required
factor	str|None	None
confidence_level	float	0.95
quartiles	Literal['haverage','tukey_hinges']	'haverage'

dependents
MinLen(min_length=1); MaxLen(max_length=50)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.509 ExploreExtreme

tabelz.analysis.report_tables_contracts.ExploreExtreme

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	str	required
rank	int	required

Field	Type	Default
case	int|None	required
value	float	required
row	str	''

4.1.510 ExploreMEstimate

tabelz.analysis.report_tables_contracts.ExploreMEstimate

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
tuning	str	required
value	float|None	required
iterations	int	required
reason	str	''

4.1.511 ExploreNormality

tabelz.analysis.report_tables_contracts.ExploreNormality

Common strict, immutable model baseline for public contracts.

Field	Type	Default
test	str	required
statistic	float|None	required
df	int	required
p_value	float|None	required
note	str	''

4.1.512 ExploreResult

tabelz.analysis.report_tables_contracts.ExploreResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	ExploreConfig	required
cells	tuple[ExploreCell,...]	required
notes	tuple[str,...]	()

4.1.513 FrequenciesConfig

tabelz.analysis.report_tables_contracts.FrequenciesConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
metadata	dict[str,ColumnMetadata]	factory:dict
treat_user_missing_as_valid	Literal[True]	True
columns	tuple[str,...]	required
percentiles	tuple[float,...]	()
quartiles	bool	False
cut_points	int|None	None
statistics	bool	True
user_missing_valid	bool	False
sets	dict[str,ResponseSet]	factory:dict

columns
MinLen(min_length=1); MaxLen(max_length=100)

cut_points
Ge(ge=2); Le(le=100)

4.1.514 FrequenciesResult

tabelz.analysis.report_tables_contracts.FrequenciesResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	FrequenciesConfig	required
tables	tuple[FrequencyTable,...]	required
notes	tuple[str,...]	()
set_tables	tuple[MrFrequenciesResult,...]	()

4.1.515 FrequencyTable

tabelz.analysis.report_tables_contracts.FrequencyTable

Common strict, immutable model baseline for public contracts.

Field	Type	Default
column	str	required
label	str	required
level	str	required
values	tuple[ValueCount,...]	required
valid	float	required
missing	float	required
statistics	tuple[tuple[str,float|None],...]	()

Field	Type	Default
percentiles	tuple[tuple[str,float|None],...]	()
histogram	tuple[tuple[float,float,int],...]	()

4.1.516 GroupStatistic

tabelz.analysis.report_tables_contracts.GroupStatistic

Common strict, immutable model baseline for public contracts.

Field	Type	Default
group	str	required
column	str	required
n	int	required
mean	float|None	required
sd	float|None	required
weight_sum	float|None	None

4.1.517 ValueCount

tabelz.analysis.report_tables_contracts.ValueCount

Common strict, immutable model baseline for public contracts.

Field	Type	Default
value	str	required
label	str	required
count	float	required
percent	float	required
valid_percent	float|None	None
cumulative_percent	float|None	None
missing	str	''

4.1.518 AnalysisResult

tabelz.analysis.results_envelope.AnalysisResult

Typed, frozen analysis-result envelope.

Field	Type	Default
schema_version	Literal['1.0.0']	'1.0.0'
status	AnalysisStatus	required
profile	RegisteredRunProfile|CustomRunProfile	required

Field	Type	Default
components	tuple[ScalarComponent|TableComponent|IntervalComponent|WarningComponent|DiagnosticComponent|ScatterPlotComponent|HistogramPlotComponent,...]	factory:tuple
sample_count	int	required
population	PopulationRef	required
operation	OperationIdentity	required

sample_count
Ge(ge=0)

4.1.519 CustomProfileIdentity

tabelz.analysis.results_envelope.CustomProfileIdentity

Declared-identity projection of a custom run profile.

Field	Type	Default
profile_kind	Literal['custom']	'custom'
settings	tuple[ProfileSetting,...]	required

4.1.520 DiagnosticComponent

tabelz.analysis.results_envelope.DiagnosticComponent

Method or engine behavior note with a stable machine code.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['diagnostic']	'diagnostic'
key	str	required
code	str	required
message	str|None	None
target_component_key	str|None	None

key
MinLen(min_length=1)
message
MinLen(min_length=1)
target_component_key
MinLen(min_length=1)

4.1.521 DiagnosticComponentFingerprint

`tabelz.analysis.results_envelope.DiagnosticComponentFingerprint`

Whitelist projection of a diagnostic; excludes display messages.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['diagnostic']	'diagnostic'
key	str	required
code	str	required
target_component_key	str|None	None

key
MinLen(min_length=1)

4.1.522 EnvelopeFingerprint

`tabelz.analysis.results_envelope.EnvelopeFingerprint`

Whitelist projection of an envelope; excludes display and storage.

Field	Type	Default
schema_version	Literal['1.0.0']	'1.0.0'
status	AnalysisStatus	required
profile	RegisteredProfileIdentity|CustomProfileIdentity	required
sample_count	int	required
population_id	str	required
table_id	str	required
operation	OperationIdentity	required
components	tuple[str,...]	factory:tuple

sample_count
Ge(ge=0)
population_id
MinLen(min_length=36); MaxLen(max_length=36)
table_id
MinLen(min_length=36); MaxLen(max_length=36)

4.1.523 HistogramPlotComponent

`tabelz.analysis.results_envelope.HistogramPlotComponent`

Bounded histogram aggregation wrapper.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['histogram_plot']	'histogram_plot'
key	str	required
payload	InlinePayload|PayloadRef	required

key
MinLen(min_length=1)

4.1.524 HistogramPlotComponentFingerprint

tabelz.analysis.results_envelope.HistogramPlotComponentFingerprint

Whitelist projection of a histogram component over its payload hash.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['histogram_plot']	'histogram_plot'
key	str	required
payload_fingerprint	str	required

key
MinLen(min_length=1)

4.1.525 IntervalComponent

tabelz.analysis.results_envelope.IntervalComponent

Confidence interval with finite bounds and an explicit method.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['interval']	'interval'
key	str	required
estimate	FloatValue	required
lower	FloatValue	required
upper	FloatValue	required
confidence_level	float	required
method	str	required
units	str|None	None
lower_inclusive	bool	required
upper_inclusive	bool	required

key
MinLen(min_length=1)
confidence_level
Gt(gt=0); Lt(lt=1)
method
MinLen(min_length=1)
units
MinLen(min_length=1)

4.1.526 IntervalComponentFingerprint

`tabelz.analysis.results_envelope.IntervalComponentFingerprint`

Whitelist projection of an interval component.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['interval']	'interval'
key	str	required
estimate	FloatValue	required
lower	FloatValue	required
upper	FloatValue	required
confidence_level	float	required
method	str	required
units	str|None	None
lower_inclusive	bool	required
upper_inclusive	bool	required

key
MinLen(min_length=1)
confidence_level
Gt(gt=0); Lt(lt=1)
method
MinLen(min_length=1)

4.1.527 OperationIdentity

`tabelz.analysis.results_envelope.OperationIdentity`

Implementation identity of an operation.

Field	Type	Default
operation_type	str	required
schema_version	str	required

Field	Type	Default
implementation_version	str	required

operation_type
MinLen(min_length=1)
schema_version
MinLen(min_length=1)
implementation_version
MinLen(min_length=1)

4.1.528 RegisteredProfileIdentity

tabelz.analysis.results_envelope.RegisteredProfileIdentity

Declared-identity projection of a registered run profile.

Field	Type	Default
profile_kind	Literal['registered']	'registered'
name	str	required
profile_version	str	required
definition_fingerprint	str	required
settings	tuple[ProfileSetting, ...]	required

name
MinLen(min_length=1)
profile_version
MinLen(min_length=1)

4.1.529 ScalarComponent

tabelz.analysis.results_envelope.ScalarComponent

One tagged numeric or scalar value with optional units and label.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['scalar']	'scalar'
key	str	required
value	IntegerValue|FloatValue|NullValue|StringValue|BooleanValue	required
units	str|None	None
label	str|None	None

key
MinLen(min_length=1)
units
MinLen(min_length=1)
label
MinLen(min_length=1)

4.1.530 ScalarComponentFingerprint

`tabelz.analysis.results_envelope.ScalarComponentFingerprint`

Whitelist projection of a scalar component; excludes display labels.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['scalar']	'scalar'
key	str	required
value	IntegerValue|FloatValue|NullValue|StringValue|BooleanValue	required
units	str|None	None

key
MinLen(min_length=1)

4.1.531 ScatterPlotComponent

`tabelz.analysis.results_envelope.ScatterPlotComponent`

Bounded scatter view wrapper with explicit x and y names.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['scatter_plot']	'scatter_plot'
key	str	required
x_name	str	required
y_name	str	required
payload	InlinePayload|PayloadRef	required

key
MinLen(min_length=1)
x_name
MinLen(min_length=1)
y_name
MinLen(min_length=1)

4.1.532 ScatterPlotComponentFingerprint

`tabelz.analysis.results_envelope.ScatterPlotComponentFingerprint`

Whitelist projection of a scatter component over its payload hash.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['scatter_plot']	'scatter_plot'
key	str	required
x_name	str	required
y_name	str	required
payload_fingerprint	str	required

key
MinLen(min_length=1)

x_name
MinLen(min_length=1)

y_name
MinLen(min_length=1)

4.1.533 TableComponent

`tabelz.analysis.results_envelope.TableComponent`

Ordered column descriptors plus a table payload wrapper.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['table']	'table'
key	str	required
columns	tuple[TableColumn, ...]	required
payload	InlinePayload|PayloadRef	required

key
MinLen(min_length=1)

columns
MinLen(min_length=1)

4.1.534 TableComponentFingerprint

`tabelz.analysis.results_envelope.TableComponentFingerprint`

Whitelist projection of a table component over its payload hash.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['table']	'table'
key	str	required
columns	tuple[TableColumn, ...]	required
payload_fingerprint	str	required

key
MinLen(min_length=1)

columns
MinLen(min_length=1)

4.1.535 WarningComponent

tabelz.analysis.results_envelope.WarningComponent

Data or population qualification with a stable machine code.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['warning']	'warning'
key	str	required
code	str	required
message	str|None	None
target_component_key	str|None	None

key
MinLen(min_length=1)

message
MinLen(min_length=1)

target_component_key
MinLen(min_length=1)

4.1.536 WarningComponentFingerprint

tabelz.analysis.results_envelope.WarningComponentFingerprint

Whitelist projection of a warning; excludes display messages.

Field	Type	Default
component_schema_version	Literal['1.0.0']	'1.0.0'
component_type	Literal['warning']	'warning'

Field	Type	Default
key	str	required
code	str	required
target_component_key	str|None	None

key
MinLen(min_length=1)

4.1.537 HistogramBinEntry

tabelz.analysis.results_payloads.HistogramBinEntry

One histogram bin with finite bounds and a non-negative count.

Field	Type	Default
lower	float	required
upper	float	required
count	int	required

count
Ge(ge=0)

4.1.538 HistogramPayloadProjection

tabelz.analysis.results_payloads.HistogramPayloadProjection

Semantic histogram payload bounded by the rendered-view bin limit.

Field	Type	Default
payload_type	Literal['histogram']	'histogram'
bins	tuple[HistogramBinEntry,...]	factory:tuple
source_count	int	required
valid_count	int	required
minimum	float|None	None
maximum	float|None	None

source_count
Ge(ge=0)
valid_count
Ge(ge=0)

4.1.539 InlinePayload

tabelz.analysis.results_payloads.InlinePayload

Payload wrapper embedding the semantic projection itself.

Field	Type	Default
storage	Literal['inline']	'inline'
payload	TablePayloadProjection|ScatterPayloadProjection|HistogramPayloadProjection	required

4.1.540 PayloadRef

tabelz.analysis.results_payloads.PayloadRef

Content-addressed durable reference to a semantic payload projection.

Field	Type	Default
storage	Literal['reference']	'reference'
payload_type	Literal['table','scatter','histogram','bar','heatmap','ecdf']	required
payload_fingerprint	str	required
canonical_byte_length	int	required
item_count	int	required
member_name	str	required

canonical_byte_length
Ge(ge=1)

item_count
Ge(ge=0)

member_name
MinLen(min_length=1)

4.1.541 ScatterPayloadProjection

tabelz.analysis.results_payloads.ScatterPayloadProjection

Semantic scatter payload bounded by the rendered-view limit.

Field	Type	Default
payload_type	Literal['scatter']	'scatter'
points	tuple[ScatterPoint,...]	factory:tuple
source_count	int	required
valid_count	int	required
rendered_count	int	required
sampled	bool	required
seed	int|None	None

source_count
Ge(ge=0)

valid_count
Ge(ge=0)
rendered_count
Ge(ge=0)

4.1.542 ScatterPoint

`tabelz.analysis.results_payloads.ScatterPoint`

One rendered scatter point with stable row identity and finite values.

Field	Type	Default
`row_id`	`str`	`required`
`x`	`float`	`required`
`y`	`float`	`required`

row_id
MinLen(min_length=1)

4.1.543 TableColumn

`tabelz.analysis.results_payloads.TableColumn`

Ordered descriptor for one tagged table column.

Field	Type	Default
`name`	`str`	`required`
`value_type`	`Literal['integer','float','null','string','boolean']`	`required`

name
MinLen(min_length=1)

4.1.544 TablePayloadProjection

`tabelz.analysis.results_payloads.TablePayloadProjection`

Semantic table payload of ordered rows of strictly typed cells.

Field	Type	Default
`payload_type`	`Literal['table']`	`'table'`
`columns`	`tuple[TableColumn,...]`	`required`
`rows`	`tuple[tuple[IntegerValue|FloatValue|NullValue|StringValue|BooleanValue,...],...]`	`factory:tuple`
`row_count`	`int`	`required`
`cell_count`	`int`	`required`

columns
MinLen(min_length=1)

row_count
Ge(ge=0)

cell_count
Ge(ge=0)

4.1.545 SeasonalAdjustmentResult

`tabelz.analysis.seasonal_adjustment.SeasonalAdjustmentResult`

Field	Type	Default
response_column	str	required
periods	tuple[int,...]	required
robust	bool	required
rows	tuple[dict[str,float|int|str],...]	required
diagnostics	dict[str,object]	required
specification	dict[str,object]	required
specification_fingerprint	str	required
data_fingerprint	str	required
reproducibility	dict[str,str]	required
warnings	tuple[str,...]	()

4.1.546 SelectionExplainerOptions

`tabelz.analysis.selection_explainer.SelectionExplainerOptions`

Complete deterministic split and ranking policy for the selection explainer.

Field	Type	Default
version	Literal['1.0.0']	'1.0.0'
validation_fraction	float	0.25
max_threshold_candidates	int	64
max_explanations	int	50
max_time_buckets	int	10
time_weight	float	1.0
numeric_weight	float	1.0
categorical_weight	float	1.0
threshold_weight	float	1.0
time_field	str|None	None
entity_key	EntityKey|None	None

validation_fraction
Gt(gt=0.0); Lt(lt=0.5)

max_threshold_candidates
Ge(ge=1); Le(le=128)
max_explanations
Ge(ge=1); Le(le=50)
max_time_buckets
Ge(ge=1); Le(le=20)
time_weight
Ge(ge=0.0)
numeric_weight
Ge(ge=0.0)
categorical_weight
Ge(ge=0.0)
threshold_weight
Ge(ge=0.0)

4.1.547 AnalysisSheet

`tabelz.analysis.sheets.AnalysisSheet`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`sheet_id`	`str`	`required`
`title`	`str`	`required`
`blocks`	`tuple[TextBlock|MethodNoteBlock|ResultTableBlock|PlotBlock|FrozenSnapshotBlock,...]`	`()`

sheet_id
MinLen(min_length=36); MaxLen(max_length=36)
title
MinLen(min_length=1)

4.1.548 AnalysisSheetCatalog

`tabelz.analysis.sheets.AnalysisSheetCatalog`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`schema_version`	`Literal['1.0.0']`	`'1.0.0'`
`sheets`	`tuple[AnalysisSheet,...]`	`()`

4.1.549 FrozenSnapshotBlock

`tabelz.analysis.sheets.FrozenSnapshotBlock`

Immutable typed capture of one rendered-result block.

Field	Type	Default
block_type	Literal['frozen_snapshot']	'frozen_snapshot'
block_id	str	required
source_block_type	Literal['result_table','plot','method_note']	required
config	SheetAnalysisConfig	required
artifact_result	ResultRef|None	None
component_key	str|None	None
details	str|None	None
state	Literal[<BlockState.FROZEN_SNAPSHOT: 'frozen_snapshot'>]	<BlockState.FROZEN_SNAPSHOT:'frozen_snapshot'>
label	Literal['Frozensnapshot']	'Frozensnapshot'

block_id
MinLen(min_length=36); MaxLen(max_length=36)

4.1.550 MethodNoteBlock

tabelz.analysis.sheets.MethodNoteBlock

Common strict, immutable model baseline for public contracts.

Field	Type	Default
block_type	Literal['method_note']	'method_note'
block_id	str	required
config	SheetAnalysisConfig	required
details	str	required
state	BlockState	<BlockState.CURRENT: 'current'>
error_message	str|None	None

block_id
MinLen(min_length=36); MaxLen(max_length=36)
details
MinLen(min_length=1)

4.1.551 PlotBlock

tabelz.analysis.sheets.PlotBlock

Common strict, immutable model baseline for public contracts.

Field	Type	Default
block_type	Literal['plot']	'plot'
block_id	str	required

Field	Type	Default
config	SheetAnalysisConfig	required
artifact_result	ResultRef|None	None
component_key	str	required
state	BlockState	<BlockState.SOURCE_CHANGED:'source_changed'>
error_message	str|None	None

block_id
MinLen(min_length=36); MaxLen(max_length=36)
component_key
MinLen(min_length=1)

4.1.552 ResultTableBlock

tabelz.analysis.sheets.ResultTableBlock

Common strict, immutable model baseline for public contracts.

Field	Type	Default
block_type	Literal['result_table']	'result_table'
block_id	str	required
config	SheetAnalysisConfig	required
artifact_result	ResultRef|None	None
component_key	str	required
state	BlockState	<BlockState.SOURCE_CHANGED:'source_changed'>
error_message	str|None	None

block_id
MinLen(min_length=36); MaxLen(max_length=36)
component_key
MinLen(min_length=1)

4.1.553 SheetAnalysisConfig

tabelz.analysis.sheets.SheetAnalysisConfig

Exact input retained so reopening can reapply the same analysis.

Field	Type	Default
operation	OperationSpec	required
profile	RegisteredRunProfile|CustomRunProfile	required

Field	Type	Default
dependency_ids	tuple[str, ...]	()

4.1.554 SheetExport

tabelz.analysis.sheets.SheetExport

Common strict, immutable model baseline for public contracts.

Field	Type	Default
format	SheetExportFormat	required
media_type	str	required
content	str	required

4.1.555 TextBlock

tabelz.analysis.sheets.TextBlock

Common strict, immutable model baseline for public contracts.

Field	Type	Default
block_type	Literal['text']	'text'
block_id	str	required
text	str	required
state	Literal[<BlockState.CURRENT: 'current'>]	<BlockState.CURRENT: 'current'>

block_id
MinLen(min_length=36); MaxLen(max_length=36)

4.1.556 BetweenWithinConfig

tabelz.analysis.spc_between_within.BetweenWithinConfig

Between/within chart settings: the within-subgroup dispersion chart is R or S.

Field	Type	Default
response	str	required
subgroup_mode	Literal['column', 'size']	'column'
subgroup_column	str|None	None
subgroup_size	int|None	None
stage_column	str|None	None
dispersion	Literal['range', 'stddev']	'range'
approved_entry	str|None	None
approved_limits	BetweenWithinApproval|None	None

response
MinLen(min_length=1)
subgroup_size
Ge(ge=2); Le(le=25)

4.1.557 BetweenWithinResult

`tabelz.analysis.spc_between_within.BetweenWithinResult`

The three stacked charts, per-stage scale estimates and the row accounting.

Field	Type	Default
response	str	required
subgroup_mode	Literal['column','size']	required
subgroup_column	str|None	required
subgroup_size	int|None	required
stage_column	str|None	required
dispersion	Literal['range','stddev']	required
subgroups	tuple[SubgroupRecord,...]	required
stages	tuple[StageScales,...]	required
individuals	ControlChartSeries	required
moving_range	ControlChartSeries	required
within	ControlChartSeries	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
dropped_subgroup_count	int	required
dropped_row_count	int	required
warnings	tuple[str,...]	()
approved_limits	BetweenWithinApproval|None	None

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
dropped_subgroup_count
Ge(ge=0)
dropped_row_count
Ge(ge=0)

4.1.558 ChartPoint

`tabelz.analysis.spc_between_within.ChartPoint`

One plotted value with its own center line and limits (“None“ when unavailable).

Field	Type	Default
subgroup	int	required
value	float	required
center	float|None	None
lower	float|None	None
upper	float|None	None
beyond_limits	bool	False

subgroup
Ge(ge=0)

4.1.559 ControlChartSeries

tabelz.analysis.spc_between_within.ControlChartSeries

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['individuals_of_means','moving_range','range','stddev']	required
title	str	required
points	tuple[ChartPoint,...]	required

4.1.560 StageScales

tabelz.analysis.spc_between_within.StageScales

The limits, scale estimates and components of one chart stage.

Field	Type	Default
index	int	required
label	str	required
first_subgroup	int	required
subgroups	int	required
observations	int	required
subgroup_size_min	int	required
subgroup_size_max	int	required
grand_mean	float	required
subgroup_mean_center	float|None	None
mean_moving_range	float	required
i_chart_scale	float	required
i_chart_available	bool	required

Field	Type	Default
within_sigma	float	required
within_estimator	str	required
within_chart_available	bool	required
components	VarianceComponents|None	required
notes	tuple[str,...]	()

index
Ge(ge=0)
first_subgroup
Ge(ge=0)
subgroups
Ge(ge=2)
observations
Ge(ge=4)
mean_moving_range
Ge(ge=0.0)
i_chart_scale
Ge(ge=0.0)
within_sigma
Ge(ge=0.0)

4.1.561 SubgroupRecord

tabelz.analysis.spc_between_within.SubgroupRecord

One analysed rational subgroup, in chart order, with its member rows.

Field	Type	Default
label	str	required
stage_index	int	required
size	int	required
mean	float	required
range	float	required
standard_deviation	float	required
row_ids	tuple[str,...]	required

stage_index
Ge(ge=0)
size
Ge(ge=2)
range
Ge(ge=0.0)

standard_deviation
Ge(ge=0.0)

4.1.562 VarianceComponents

`tabelz.analysis.spc_between_within.VarianceComponents`

Components of the individual observations from one REML random-intercept fit.

Field	Type	Default
df_between	int	required
df_within	int	required
ms_between	float	required
ms_within	float	required
n0	float	required
between_variance	float	required
between_truncated	bool	required
within_variance	float	required
total_variance	float	required
between_sigma	float	required
within_component_sigma	float	required
total_sigma	float	required
estimator	str	'one-wayrandom-effectsANOVAestimator(methodofmoments)'
fit_method	str	''

df_between
Ge(ge=1)
df_within
Ge(ge=1)
ms_between
Ge(ge=0.0)
ms_within
Ge(ge=0.0)
n0
Gt(gt=0.0)
between_variance
Ge(ge=0.0)
within_variance
Ge(ge=0.0)
total_variance
Ge(ge=0.0)
between_sigma
Ge(ge=0.0)

within_component_sigma
Ge(ge=0.0)
total_sigma
Ge(ge=0.0)

4.1.563 BetweenWithinApproval

`tabelz.analysis.spc_subgroup_approval.BetweenWithinApproval`

Approved limits of the between/within I-MR-R/S chart.

Field	Type	Default
entry_id	str	required
dataset_id	str	required
characteristic	str	required
process_stage	str	''
revision	str	required
approved_by	str	required
approved_at	str	required
rationale	str	''
baseline_operation	str	required
baseline_implementation_version	str	required
subgroup_mode	Literal['column','size']	required
subgroup_column	str|None	None
subgroup_size	int|None	None
baseline_subgroups	int	required
subgroup_size_min	int	required
subgroup_size_max	int	required
chart	Literal['between_within']	'between_within'
center	float	required
mean_moving_range	float	required
dispersion	Literal['range','stddev']	required
within_sigma	float	required
within_estimator	Literal['R-bar/d2(n)','S-bar/c4(n)']	required
rules	tuple[str,...]	('rule_1',)

entry_id
MinLen(min_length=1); MaxLen(max_length=64)
dataset_id
MinLen(min_length=1)
characteristic
MinLen(min_length=1)
revision
MinLen(min_length=1)

approved_by
MinLen(min_length=1)
approved_at
MinLen(min_length=1)
baseline_operation
MinLen(min_length=1)
baseline_implementation_version
MinLen(min_length=1)
baseline_subgroups
Ge(ge=2)
subgroup_size_min
Ge(ge=2)
subgroup_size_max
Ge(ge=2)
mean_moving_range
Gt(gt=0.0)
within_sigma
Gt(gt=0.0)

4.1.564 ZoneApproval

`tabelz.analysis.spc_subgroup_approval.ZoneApproval`

Approved centre, scale and scoring settings of the zone chart.

Field	Type	Default
entry_id	str	required
dataset_id	str	required
characteristic	str	required
process_stage	str	''
revision	str	required
approved_by	str	required
approved_at	str	required
rationale	str	''
baseline_operation	str	required
baseline_implement ation_version	str	required
subgroup_mode	Literal['column','size']	required
subgroup_column	str|None	None
subgroup_size	int|None	None
baseline_subgroups	int	required
subgroup_size_min	int	required
subgroup_size_max	int	required
chart	Literal['zone']	'zone'

Field	Type	Default
`statistic`	`Literal['subgroup_mean','individuals_of_means']`	required
`within_sigma_estimator`	`Literal['rbar','sbar']`	required
`center`	`float`	required
`sigma`	`float`	required
`sigma_description`	`str`	required
`weights`	`tuple[float,...]`	required
`signal_score`	`float`	required
`reset_after_signal`	`bool`	required

entry_id
MinLen(min_length=1); MaxLen(max_length=64)
dataset_id
MinLen(min_length=1)
characteristic
MinLen(min_length=1)
revision
MinLen(min_length=1)
approved_by
MinLen(min_length=1)
approved_at
MinLen(min_length=1)
baseline_operation
MinLen(min_length=1)
baseline_implementation_version
MinLen(min_length=1)
baseline_subgroups
Ge(ge=2)
subgroup_size_min
Ge(ge=2)
subgroup_size_max
Ge(ge=2)
sigma
Gt(gt=0.0)
signal_score
Gt(gt=0.0)

4.1.565 ZoneChartConfig

`tabelz.analysis.spc_zone.ZoneChartConfig`

Zone chart settings; the weights and signal value are recorded with the result.

Field	Type	Default
response	str	required
subgroup_mode	Literal['column','size']	'column'
subgroup_column	str|None	None
subgroup_size	int|None	None
stage_column	str|None	None
statistic	Literal['subgroup_mean','individuals_of_means']	'subgroup_mean'
within_sigma	Literal['rbar','sbar']	'rbar'
weights	tuple[float,...]	(0.0,2.0,4.0,8.0)
signal_score	float	8.0
reset_after_signal	bool	False
approved_entry	str|None	None
approved_limits	ZoneApproval|None	None

response
MinLen(min_length=1)

subgroup_size
Ge(ge=2); Le(le=25)

4.1.566 ZoneChartResult

tabelz.analysis.spc_zone.ZoneChartResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
subgroup_mode	Literal['column','size']	required
subgroup_column	str|None	required
subgroup_size	int|None	required
stage_column	str|None	required
statistic	Literal['subgroup_mean','individuals_of_means']	required
within_sigma	Literal['rbar','sbar']	required
weights	tuple[float,...]	required
signal_score	float	required
reset_after_signal	bool	required
customized_signal	bool	required
subgroups	tuple[SubgroupRecord,...]	required
stages	tuple[ZoneStage,...]	required
points	tuple[ZonePoint,...]	required
signal_count	int	required
source_count	int	required

Field	Type	Default
excluded_count	int	required
missing_count	int	required
dropped_subgroup_count	int	required
dropped_row_count	int	required
warnings	tuple[str, ...]	()
approved_limits	ZoneApproval|None	None

signal_count
Ge(ge=0)
source_count
Ge(ge=0)
excluded_count
Ge(ge=0)
missing_count
Ge(ge=0)
dropped_subgroup_count
Ge(ge=0)
dropped_row_count
Ge(ge=0)

4.1.567 ZonePoint

tabelz.analysis.spc_zone.ZonePoint

Common strict, immutable model baseline for public contracts.

Field	Type	Default
subgroup	int	required
stage_index	int	required
value	float	required
center	float|None	None
scale	float|None	None
z	float|None	None
zone	int|None	None
side	int	0
weight	float|None	None
score	float|None	None
signal	bool	False
event	Literal['stage_start','side_change', 'after_reset','center_line','']	''

subgroup
Ge(ge=0)

stage_index
Ge(ge=0)
zone
Ge(ge=1); Le(le=4)
side
Ge(ge=-1); Le(le=1)

4.1.568 ZoneStage

`tabelz.analysis.spc_zone.ZoneStage`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`index`	`int`	`required`
`label`	`str`	`required`
`first_subgroup`	`int`	`required`
`subgroups`	`int`	`required`
`center`	`float`	`required`
`sigma`	`float`	`required`
`sigma_description`	`str`	`required`
`available`	`bool`	`required`
`unavailable_reason`	`str`	`''`
`signals`	`int`	`0`

index
Ge(ge=0)
first_subgroup
Ge(ge=0)
subgroups
Ge(ge=2)
sigma
Ge(ge=0.0)
signals
Ge(ge=0)

4.1.569 ResolvedSpecification

`tabelz.analysis.specifications.ResolvedSpecification`

The unique applicable specification together with its audit receipt.

Field	Type	Default
`specification`	`Specification`	`required`
`receipt`	`SpecificationReceipt`	`required`

4.1.570 Specification

`tabelz.analysis.specifications.Specification`

One immutable, schema-versioned limit/target record and its applicability.

Field	Type	Default
`key`	`str`	`required`
`revision`	`str`	`required`
`units`	`str`	`required`
`lower`	`float|None`	`None`
`upper`	`float|None`	`None`
`target`	`float|None`	`None`
`effective_from`	`datetime.date|None`	`None`
`effective_to`	`datetime.date|None`	`None`
`applicability`	`tuple[tuple[str,str],...]`	`()`
`schema_version`	`str`	`'1.0.0'`

4.1.571 SpecificationReceipt

`tabelz.analysis.specifications.SpecificationReceipt`

Canonical evidence identifying the resolved record and any unit conversion.

Field	Type	Default
`schema_version`	`str`	`required`
`key`	`str`	`required`
`revision`	`str`	`required`
`units`	`str`	`required`
`lower`	`float|None`	`required`
`upper`	`float|None`	`required`
`target`	`float|None`	`required`
`effective_from`	`str|None`	`required`
`effective_to`	`str|None`	`required`
`as_of`	`str|None`	`required`
`applicability`	`tuple[tuple[str,str],...]`	`required`
`requested_units`	`str`	`required`
`conversion_factor`	`float`	`required`
`conversion_offset`	`float`	`required`
`fingerprint`	`str`	`required`

4.1.572 OverlaidContourConfig

`tabelz.analysis.stored_model_contour.OverlaidContourConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
profiler	StoredProfilerConfig	required
x	str	required
y	str	required
grid	int	41

grid
Ge(ge=11); Le(le=101)

4.1.573 OverlaidContourResult

tabelz.analysis.stored_model_contour.OverlaidContourResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	OverlaidContourConfig	required
x_nodes	tuple[float,...]	required
y_nodes	tuple[float,...]	required
held_settings	dict[str,float|str]	required
responses	tuple[OverlaidResponse,...]	required
feasible	tuple[tuple[bool,...],...]	required
feasible_fraction	float	required

4.1.574 OverlaidResponse

tabelz.analysis.stored_model_contour.OverlaidResponse

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response_name	str	required
goal	Literal['maximize','minimize','target']	required
low	float	required
high	float	required
z_grid	tuple[tuple[float,...],...]	required
z_range	tuple[float,float]	required
acceptable_fraction	float	required

4.1.575 ProfilerModelBinding

tabelz.analysis.stored_model_profiler.ProfilerModelBinding

Common strict, immutable model baseline for public contracts.

Field	Type	Default
artifact	ModelScoringArtifact|DoeModelArtifact	required
exposure	float	1.0
fitted_ranges	dict[str,tuple[float,float]]	factory:dict

exposure
Gt(gt=0.0); Le(le=1000000000.0)

4.1.576 StoredProfilerConfig

tabelz.analysis.stored_model_profiler.StoredProfilerConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
models	tuple[ProfilerModelBinding,...]	required
predictors	tuple[PredictorSpecification,...]	required
goals	tuple[DesirabilityGoal,...]	required
confidence_level	float	0.95
seed	int	42
search_samples	int	5000
local_evaluations	int	1024

models
MinLen(min_length=1); MaxLen(max_length=8)
predictors
MinLen(min_length=1); MaxLen(max_length=16)
confidence_level
Gt(gt=0.5); Lt(lt=1.0)
seed
Ge(ge=0); Le(le=9223372036854775807)
search_samples
Ge(ge=128); Le(le=10000)
local_evaluations
Ge(ge=0); Le(le=4096)

4.1.577 ZivotAndrewsResult

tabelz.analysis.structural_breaks.ZivotAndrewsResult

Field	Type	Default
response_column	str	required
deterministic_term	str	required
statistic	float	required
p_value	float	required

Field	Type	Default
break_index	int	required
break_label	str	required
selected_lags	int	required
critical_values	dict[str,float]	required
interpretation	str	required
points	tuple[dict[str,object],...]	required
specification	dict[str,object]	required
specification_fingerprint	str	required
data_fingerprint	str	required
reproducibility	dict[str,str]	required
warnings	tuple[str,...]	()

4.1.578 SurveyCalibrationConfig

tabelz.analysis.survey_calibration_contracts.SurveyCalibrationConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
design	SurveyDesign	required
method	Literal['poststratification','raking','linear']	'poststratification'
targets	tuple[SurveyCalibrationTarget,...]	required
prefix	str	'Calibrated'
lower_bound	float|None	None
upper_bound	float|None	None
tolerance	float	1e-09
max_iterations	int	100

targets
MinLen(min_length=1); MaxLen(max_length=50)
prefix
MinLen(min_length=1); MaxLen(max_length=64)
lower_bound
Gt(gt=0)
upper_bound
Gt(gt=0)
tolerance
Gt(gt=0); Le(le=0.01)
max_iterations
Ge(ge=1); Le(le=1000)

4.1.579 SurveyCalibrationResult

`tabelz.analysis.survey_calibration_contracts.SurveyCalibrationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`SurveyCalibrationConfig`	`required`
`calibrated_design`	`SurveyDesign`	`required`
`rows`	`int`	`required`
`constraint_rank`	`int`	`required`
`calibrated_replicates`	`int`	`required`
`maximum_relative_error`	`float`	`required`
`minimum_factor`	`float`	`required`
`maximum_factor`	`float`	`required`
`target_estimates`	`tuple[float, ...]`	`required`
`fingerprint`	`str`	`required`
`notes`	`tuple[str, ...]`	`required`
`warnings`	`tuple[str, ...]`	`()`
`method_version`	`Literal['1.0.0']`	`'1.0.0'`

4.1.580 RaoScottTest

`tabelz.analysis.survey_contracts.RaoScottTest`

Second-order Rao-Scott test of independence (the R “svychisq“ F statistic).

Field	Type	Default
`pearson`	`float`	`required`
`trace`	`float`	`required`
`trace_squared`	`float`	`required`
`mean_design_effect`	`float`	`required`
`f_statistic`	`float`	`required`
`numerator_df`	`float`	`required`
`denominator_df`	`float`	`required`
`p_value`	`float`	`required`
`contingency_df`	`int`	`required`

4.1.581 SurveyCell

`tabelz.analysis.survey_contracts.SurveyCell`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row	str	required
column	str|None	None
rows	int	required
count	float	required
count_se	float	required
percent	float	required
percent_se	float	required
percent_lower	float	required
percent_upper	float	required
row_percent	float|None	None
row_percent_se	float|None	None
column_percent	float|None	None
column_percent_se	float|None	None

4.1.582 SurveyCrosstabConfig

tabelz.analysis.survey_contracts.SurveyCrosstabConfig

Weighted frequencies of one column, or a crosstab of two with the Rao-Scott test.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()
confidence_level	float	0.95
row_column	str	required
column_column	str|None	None

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

row_column
MinLen(min_length=1)

4.1.583 SurveyCrosstabResult

tabelz.analysis.survey_contracts.SurveyCrosstabResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyCrosstabConfig	required
design	SurveyDesignSummary	required
row_levels	tuple[str,...]	required

Field	Type	Default
column_levels	tuple[str,...]	()
cells	tuple[SurveyCell,...]	required
analysis_rows	int	required
missing_rows	int	required
test	RaoScottTest|None	None
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
method_version	str	'1.0.0'

4.1.584 SurveyEstimate

tabelz.analysis.survey_contracts.SurveyEstimate

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variable	str	required
level	str|None	None
statistic	Literal['Mean','Total','Proportion', 'Count']	required
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required
df	int	required
rows	int	required
missing_rows	int	required
extensions	EstimateExtensions|None	None

4.1.585 SurveyEstimatesConfig

tabelz.analysis.survey_contracts.SurveyEstimatesConfig

Means and totals of numeric columns; proportions and counts of categorical ones.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()
confidence_level	float	0.95
numeric_columns	tuple[str,...]	()
categorical_columns	tuple[str,...]	()

Field	Type	Default
design_effect	Literal['none','srs_wr','srs_wor']|None	None
bounded_interval	Literal['none','delta_logit','korn_graubard']|None	None
weights_population_scale	bool|None	None

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

numeric_columns
MaxLen(max_length=50)

categorical_columns
MaxLen(max_length=50)

4.1.586 SurveyEstimatesResult

tabelz.analysis.survey_contracts.SurveyEstimatesResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyEstimatesConfig	required
design	SurveyDesignSummary	required
estimates	tuple[SurveyEstimate,...]	required
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
method_version	str	'1.0.0'

4.1.587 SurveyRatio

tabelz.analysis.survey_contracts.SurveyRatio

Common strict, immutable model baseline for public contracts.

Field	Type	Default
numerator	str	required
denominator	str	required
ratio	float	required
standard_error	float	required
lower	float	required
upper	float	required
df	int	required
rows	int	required

Field	Type	Default
missing_rows	int	required

4.1.588 SurveyRatioConfig

tabelz.analysis.survey_contracts.SurveyRatioConfig

What every complex-samples procedure is given.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()
confidence_level	float	0.95
numerators	tuple[str,...]	required
denominator	str	required

confidence_level
Gt(gt=0.5); Lt(lt=1.0)
numerators
MinLen(min_length=1); MaxLen(max_length=50)
denominator
MinLen(min_length=1)

4.1.589 SurveyRatioResult

tabelz.analysis.survey_contracts.SurveyRatioResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyRatioConfig	required
design	SurveyDesignSummary	required
ratios	tuple[SurveyRatio,...]	required
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
method_version	str	'1.0.0'

4.1.590 SurveyCountConfig

tabelz.analysis.survey_count_contracts.SurveyCountConfig

Predictors shared by the design-based models: factors (reference coded) and covariates.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()
confidence_level	float	0.95
factors	tuple[str,...]	()
covariates	tuple[str,...]	()
interactions	tuple[tuple[str,...],...]|None	None
response	str	required
inference	Literal['design_df','normal']	'design_df'
family	Literal['poisson','nb2_fixed','nb2_estimated']	'poisson'
dispersion	float|None	None
exposure	str|None	None

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

factors
MaxLen(max_length=30)

covariates
MaxLen(max_length=30)

interactions
MaxLen(max_length=30)

response
MinLen(min_length=1)

dispersion
Gt(gt=0)

4.1.591 SurveyCountResult

tabelz.analysis.survey_count_contracts.SurveyCountResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyCountConfig	required
fit	SurveyModelFit	required
dispersion	float|None	None
dispersion_standard_error	float|None	None
dispersion_uncertainty	str	required

4.1.592 SurveyDescriptivesConfig

tabelz.analysis.survey_descriptives.SurveyDescriptivesConfig

Immutable request for one design based descriptive estimate.

Field	Type	Default
response_column	str	required
weight_column	str	required
stratum_column	str	required
psu_column	str	required
mode	Literal['mean','proportion']	'mean'
confidence_level	float	0.95
max_rows	int	1000000

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.593 SurveyDescriptivesResult

tabelz.analysis.survey_descriptives.SurveyDescriptivesResult

JSON-safe survey estimate and design accounting.

Field	Type	Default
response_column	str	required
weight_column	str	required
stratum_column	str	required
psu_column	str	required
mode	Literal['mean','proportion']	required
confidence_level	float	required
estimate	float	required
standard_error	float	required
lower	float	required
upper	float	required
degrees_of_freedom	int	required
source_count	int	required
observations	int	required
missing_count	int	required
excluded_count	int	required
weight_sum	float	required
stratum_count	int	required
psu_count	int	required
warnings	tuple[str,...]	()

Field	Type	Default
method	str	'survey_descriptives.one_stage_wr_taylor_ratio'
implementation_version	str	'1.0.0'
retained_row_ids	tuple[str,...]	()
population_fingerprint	str|None	None

4.1.594 SurveyDesignSummary

tabelz.analysis.survey_design.SurveyDesignSummary

What every complex-samples result discloses about the design it used.

Field	Type	Default
variance_method	str	required
weight	str	required
rows	int	required
weight_total	float	required
strata	int|None	None
psus	int|None	None
replicates	int|None	None
design_df	int	required
design_df_rule	str	required
stage1_fpc	str	required
later_stage_variance	str	required
single_psu	str	required
combined_strata	tuple[str,...]	()
combination_justification	str	''
certainty_strata	tuple[str,...]	()
domain	str	'Allrows'
domain_rows	int	0
domain_weight_total	float	0.0
notes	tuple[str,...]	()

4.1.595 EstimateExtensions

tabelz.analysis.survey_estimate_extensions.EstimateExtensions

Common strict, immutable model baseline for public contracts.

Field	Type	Default
comparator	str	required
actual_sample_n	int	required
srs_variance	float|None	None
design_effect	float|None	None
deft_wr	float|None	None
design_effect_note	str	required
bounded_method	str	required
bounded_lower	float|None	None
bounded_upper	float|None	None
effective_n	float|None	None
beta_df_sample_n	int|None	None
bounded_note	str	required

4.1.596 SurveyCoefficient

tabelz.analysis.survey_models_contracts.SurveyCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
estimate	float	required
standard_error	float	required
statistic	float	required
df	int|None	required
p_value	float	required
lower	float	required
upper	float	required
ratio	float|None	None
ratio_lower	float|None	None
ratio_upper	float|None	None

4.1.597 SurveyCoxConfig

tabelz.analysis.survey_models_contracts.SurveyCoxConfig

Design-based Cox proportional hazards regression (Efron ties, Binder variance).

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()

Field	Type	Default
confidence_level	float	0.95
factors	tuple[str,...]	()
covariates	tuple[str,...]	()
interactions	tuple[tuple[str,...],...]|None	None
time	str	required
status	str	required
event_value	str	required
entry	str|None	None
subject	str|None	None
baseline_strata	str|None	None

confidence_level
Gt(gt=0.5); Lt(lt=1.0)
factors
MaxLen(max_length=30)
covariates
MaxLen(max_length=30)
interactions
MaxLen(max_length=30)
time
MinLen(min_length=1)
status
MinLen(min_length=1)
event_value
MinLen(min_length=1)
entry
MinLen(min_length=1)
subject
MinLen(min_length=1)
baseline_strata
MinLen(min_length=1)

4.1.598 SurveyCoxResult

tabelz.analysis.survey_models_contracts.SurveyCoxResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyCoxConfig	required
events	int	required
fit	SurveyModelFit	required

4.1.599 SurveyLinearConfig

`tabelz.analysis.survey_models_contracts.SurveyLinearConfig`

Design-based linear regression (weighted least squares, linearized variance).

Field	Type	Default
`design`	`SurveyDesign`	`required`
`domain_column`	`str|None`	`None`
`domain_values`	`tuple[str,...]`	`()`
`confidence_level`	`float`	`0.95`
`factors`	`tuple[str,...]`	`()`
`covariates`	`tuple[str,...]`	`()`
`interactions`	`tuple[tuple[str,...],...]|None`	`None`
`response`	`str`	`required`
`inference`	`Literal['design_df','normal']`	`'design_df'`

confidence_level
Gt(gt=0.5); Lt(lt=1.0)
factors
MaxLen(max_length=30)
covariates
MaxLen(max_length=30)
interactions
MaxLen(max_length=30)
response
MinLen(min_length=1)

4.1.600 SurveyLinearResult

`tabelz.analysis.survey_models_contracts.SurveyLinearResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`SurveyLinearConfig`	`required`
`fit`	`SurveyModelFit`	`required`

4.1.601 SurveyLogisticConfig

`tabelz.analysis.survey_models_contracts.SurveyLogisticConfig`

Design-based binary logistic regression by pseudo-maximum likelihood.

Field	Type	Default
`design`	`SurveyDesign`	`required`
`domain_column`	`str|None`	`None`

Field	Type	Default
domain_values	tuple[str,...]	()
confidence_level	float	0.95
factors	tuple[str,...]	()
covariates	tuple[str,...]	()
interactions	tuple[tuple[str,...],...]|None	None
response	str	required
inference	Literal['design_df','normal']	'design_df'
event_value	str	required

confidence_level
Gt(gt=0.5); Lt(lt=1.0)
factors
MaxLen(max_length=30)
covariates
MaxLen(max_length=30)
interactions
MaxLen(max_length=30)
response
MinLen(min_length=1)
event_value
MinLen(min_length=1)

4.1.602 SurveyLogisticResult

tabelz.analysis.survey_models_contracts.SurveyLogisticResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyLogisticConfig	required
fit	SurveyModelFit	required

4.1.603 SurveyModelFit

tabelz.analysis.survey_models_contracts.SurveyModelFit

What every design-based model reports besides its coefficients.

Field	Type	Default
design	SurveyDesignSummary	required
rows_used	int	required
rows_missing	int	required
rank	int	required
design_df	int	required

Field	Type	Default
test_df	int|None	required
reference	Literal['t','normal']	required
iterations	int	required
coefficients	tuple[SurveyCoefficient,...]	required
thresholds	tuple[SurveyCoefficient,...]	()
term_tests	tuple[SurveyTermTest,...]	()
statistics	tuple[SurveyStatistic,...]	()
notes	tuple[str,...]	()
warnings	tuple[str,...]	()
method_version	str	'1.0.0'

4.1.604 SurveyOrdinalConfig

tabelz.analysis.survey_models_contracts.SurveyOrdinalConfig

Design-based proportional-odds (cumulative logit) regression.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str,...]	()
confidence_level	float	0.95
factors	tuple[str,...]	()
covariates	tuple[str,...]	()
interactions	tuple[tuple[str,...],...]|None	None
response	str	required
level_order	tuple[str,...]	()

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

factors
MaxLen(max_length=30)

covariates
MaxLen(max_length=30)

interactions
MaxLen(max_length=30)

response
MinLen(min_length=1)

4.1.605 SurveyOrdinalResult

tabelz.analysis.survey_models_contracts.SurveyOrdinalResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	SurveyOrdinalConfig	required
levels	tuple[str, ...]	required
fit	SurveyModelFit	required

4.1.606 SurveyStatistic

tabelz.analysis.survey_models_contracts.SurveyStatistic

Common strict, immutable model baseline for public contracts.

Field	Type	Default
label	str	required
value	float	required

4.1.607 SurveyTermTest

tabelz.analysis.survey_models_contracts.SurveyTermTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
df	int	required
statistic	float	required
denominator_df	int|None	required
p_value	float	required

4.1.608 SurveyMultinomialConfig

tabelz.analysis.survey_multinomial_contracts.SurveyMultinomialConfig

Predictors shared by the design-based models: factors (reference coded) and covariates.

Field	Type	Default
design	SurveyDesign	required
domain_column	str|None	None
domain_values	tuple[str, ...]	()
confidence_level	float	0.95
factors	tuple[str, ...]	()
covariates	tuple[str, ...]	()
interactions	tuple[tuple[str, ...], ...]|None	None
response	str	required
reference_value	str	required

confidence_level
Gt(gt=0.5); Lt(lt=1.0)
factors
MaxLen(max_length=30)
covariates
MaxLen(max_length=30)
interactions
MaxLen(max_length=30)
response
MinLen(min_length=1)
reference_value
MinLen(min_length=1)

4.1.609 SurveyMultinomialResult

`tabelz.analysis.survey_multinomial_contracts.SurveyMultinomialResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`SurveyMultinomialConfig`	`required`
`levels`	`tuple[str, ...]`	`required`
`fit`	`SurveyModelFit`	`required`

4.1.610 SurveyReplicateConfig

`tabelz.analysis.survey_weight_contracts.SurveyReplicateConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`design`	`SurveyDesign`	`required`
`method`	`Literal['jk1','jkn','brr','fay','rao_wu']`	`'jkn'`
`prefix`	`str`	`'Replicate'`
`assume_srs_psu`	`bool`	`False`
`replicates`	`int`	`100`
`fay_rho`	`float`	`0.5`
`seed`	`int`	`970149`
`centering`	`Literal['full_sample','replicate_mean']`	`'full_sample'`

prefix
MinLen(min_length=1); MaxLen(max_length=64)
replicates
Ge(ge=2); Le(le=500)

fay_rho
Gt(gt=0); Lt(lt=1)
seed
Ge(ge=0); Le(le=4294967295)

4.1.611 SurveyReplicateResult

`tabelz.analysis.survey_weight_contracts.SurveyReplicateResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`config`	`SurveyReplicateConfig`	`required`
`generated_design`	`SurveyDesign`	`required`
`rows`	`int`	`required`
`random_psus`	`int`	`required`
`random_strata`	`int`	`required`
`replicate_count`	`int`	`required`
`degrees_of_freedom`	`int`	`required`
`fingerprint`	`str`	`required`
`minimum_weight`	`float`	`required`
`maximum_weight`	`float`	`required`
`notes`	`tuple[str,...]`	`required`
`warnings`	`tuple[str,...]`	`()`
`method_version`	`Literal['1.0.0']`	`'1.0.0'`

4.1.612 LogRankTest

`tabelz.analysis.survival_nonparametric.LogRankTest`

Two-group log-rank test, or the explicit reason it is unavailable.

Field	Type	Default
`status`	`Literal['computed','unavailable']`	`required`
`groups`	`tuple[str,...]`	`required`
`observed`	`tuple[int,...]`	`required`
`expected`	`tuple[float,...]|None`	`None`
`variance`	`float|None`	`None`
`statistic`	`float|None`	`None`
`degrees_of_freedom`	`int`	`1`
`p_value`	`float|None`	`None`
`reason`	`str|None`	`None`

4.1.613 MedianSurvival

`tabelz.analysis.survival_nonparametric.MedianSurvival`

Median survival time and its band-inversion confidence limits.

Field	Type	Default
status	Literal['estimated','not_reached']	required
time	float|None	required
lower_status	Literal['estimated','not_reached','unavailable']	required
lower	float|None	required
upper_status	Literal['estimated','not_reached','unavailable']	required
upper	float|None	required

4.1.614 SurvivalConfig

tabelz.analysis.survival_nonparametric.SurvivalConfig

Frozen, strict Kaplan-Meier request.

Field	Type	Default
time_column	str	required
event_column	str	required
event_value	str	required
censored_value	str	required
group_column	str|None	None
confidence_level	float	0.95
max_rows	int	1000000

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

4.1.615 SurvivalCurve

tabelz.analysis.survival_nonparametric.SurvivalCurve

Exact Kaplan-Meier table for one group, one entry per distinct time.

Field	Type	Default
group	str|None	required
n	int	required
events	int	required
censored	int	required
times	tuple[float,...]	required
at_risk	tuple[int,...]	required

Field	Type	Default
event_counts	tuple[int,...]	required
censor_counts	tuple[int,...]	required
survival	tuple[float,...]	required
standard_error	tuple[float|None,...]	required
lower	tuple[float|None,...]	required
upper	tuple[float|None,...]	required
median	MedianSurvival	required
inference_available	bool	required

n
Ge(ge=1)
events
Ge(ge=0)
censored
Ge(ge=0)

4.1.616 SurvivalResult

tabelz.analysis.survival_nonparametric.SurvivalResult

Frozen, strict grouped Kaplan-Meier outcome for transport and rendering.

Field	Type	Default
time_column	str	required
event_column	str	required
event_value	str	required
censored_value	str	required
group_column	str|None	required
confidence_level	float	required
curves	tuple[SurvivalCurve,...]	required
log_rank	LogRankTest|None	required
source_count	int	required
excluded_count	int	required
missing_count	int	required
analyzed_count	int	required
population_fingerprint	str	required
warnings	tuple[str,...]	()

source_count
Ge(ge=0)
excluded_count
Ge(ge=0)

missing_count
Ge(ge=0)
analyzed_count
Ge(ge=1)

4.1.617 TabulationCell

`tabelz.analysis.tabulation.TabulationCell`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
row_value	str	required
column_value	str|None	None
count	int	required
row_total	int	required
column_total	int	required
overall_total	int	required
row_percentage	float|None	None
column_percentage	float|None	None
overall_percentage	float|None	None
is_margin	bool	False

4.1.618 TabulationConfig

`tabelz.analysis.tabulation.TabulationConfig`

Strict request for grouped numeric summaries or a one/two-way table.

Field	Type	Default
kind	Literal['numeric','categorical']	'numeric'
value_column	str|None	None
group_columns	tuple[str,...]	()
row_column	str|None	None
column_column	str|None	None
frequency_column	str|None	None
row_percentages	bool	False
column_percentages	bool	False
overall_percentages	bool	False
include_margins	bool	True
confidence_level	float	0.95
max_rows	int	1000000
max_cells	int	2000000
treat_user_missing_as_valid	bool	False

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

max_rows
Ge(ge=1); Le(le=1000000)

max_cells
Ge(ge=1); Le(le=2000000)

4.1.619 TabulationNumericSummary

`tabelz.analysis.tabulation.TabulationNumericSummary`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
group_values	tuple[str,...]	required
count	int	required
valid_n	int	required
missing_n	int	required
mean	float|None	required
standard_deviation	float|None	required
standard_error	float|None	required
confidence_lower	float|None	required
confidence_upper	float|None	required
median	float|None	required
first_quartile	float|None	required
third_quartile	float|None	required
minimum	float|None	required
maximum	float|None	required
range	float|None	required

4.1.620 TabulationResult

`tabelz.analysis.tabulation.TabulationResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['numeric','categorical']	required
value_column	str|None	required
group_columns	tuple[str,...]	required
row_column	str|None	required
column_column	str|None	required
frequency_column	str|None	required
confidence_level	float	required
source_count	int	required

Field	Type	Default
observations	int	required
valid_n	int	required
missing_count	int	required
missing_n	int	required
excluded_count	int	required
numeric_summaries	tuple[TabulationNumericSummary,...]	()
cells	tuple[TabulationCell,...]	()
warnings	tuple[str,...]	()
retained_row_ids	tuple[str,...]	()
population_fingerprint	str	required
method	str	'tabulation.grouped_descriptives_crosstab'
implementation_version	str	'1.0.0'

4.1.621 AdvancedTimeSeriesSpec

tabelz.analysis.time_series_advanced.AdvancedTimeSeriesSpec

Replayable specification for one advanced time-series analysis.

Field	Type	Default
kind	Literal['garch','var_vecm','zivot_andrews','seasonal_adjustment']	required
columns	tuple[str,...]	required
time_column	str|None	None
horizon	int	12
confidence_level	float	0.95
garch_p	int	1
garch_q	int	1
distribution	Literal['normal','t']	't'
mean	Literal['constant','zero']	'constant'
lag_order	int	1
cointegration_rank	int|None	None
deterministic	Literal['none','constant','trend']	'constant'
break_deterministic	Literal['constant','trend','both']	'both'
periods	tuple[int,...]	()
robust	bool	True
simulation_draws	int	500
random_seed	int	0

Field	Type	Default
units	tuple[str,...]	()
calendar_name	str|None	None
timezone	str	'UTC'
schema_version	str	'1.0.0'

4.1.622 AdvancedTimeSeriesStudySpec

tabelz.analysis.time_series_advanced.AdvancedTimeSeriesStudySpec

Saved rolling validation specification for GARCH or VAR/VECM.

Field	Type	Default
analysis	AdvancedTimeSeriesSpec	required
validation	AdvancedValidationConfig	required
study_name	str	'Advancedtime-series study'
schema_version	str	'1.0.0'

4.1.623 AdvancedValidationConfig

tabelz.analysis.time_series_advanced.AdvancedValidationConfig

Rolling-origin design for advanced forecast models.

Field	Type	Default
initial_window	int	required
horizon	int	1
step	int	1
window	Literal['expanding','sliding']	'expanding'
sliding_window	int|None	None
final_evaluation	int	1

4.1.624 DifferencingStage

tabelz.analysis.time_series_contracts.DifferencingStage

Field	Type	Default
ordinary_order	int	required
sample_count	int	required
stationarity	StationarityDiagnostics	required

4.1.625 ForecastCandidate

`tabelz.analysis.time_series_contracts.ForecastCandidate`

Field	Type	Default
`model`	`str`	`required`
`label`	`str`	`required`
`status`	`Literal['ok','warning','failed']`	`required`
`metrics`	`ForecastMetrics|None`	`None`
`aic`	`float|None`	`None`
`bic`	`float|None`	`None`
`message`	`str|None`	`None`
`converged`	`bool|None`	`None`

4.1.626 ForecastConfig

`tabelz.analysis.time_series_contracts.ForecastConfig`

Complete effective settings for one univariate forecast.

Field	Type	Default
`response_column`	`str`	`required`
`time_column`	`str|None`	`None`
`model`	`Literal['automatic','auto_arima','naive','seasonal_naive','mean','drift','ses','holt','damped_holt','holt_winters_additive','arima','sarima']`	`'automatic'`
`horizon`	`int`	`12`
`holdout`	`int`	`0`
`confidence_level`	`float`	`0.95`
`season_length`	`int|None`	`None`
`arima_order`	`tuple[int,int,int]`	`(1,1,1)`
`seasonal_order`	`tuple[int,int,int]`	`(1,1,1)`
`missing_policy`	`Literal['error','drop']`	`'error'`
`calendar`	`Literal['strict','weekdays']`	`'strict'`
`deterministic_term`	`Literal['automatic','none','constant','drift']`	`'automatic'`
`stationarity_term`	`Literal['constant','trend']`	`'constant'`

4.1.627 ForecastMetrics

`tabelz.analysis.time_series_contracts.ForecastMetrics`

Field	Type	Default
mae	float	required
rmse	float	required
mase	float|None	required
smape	float	required
bias	float	required
interval_coverage	float|None	required

4.1.628 ForecastParameter

tabelz.analysis.time_series_contracts.ForecastParameter

Field	Type	Default
term	str	required
estimate	float	required
standard_error	float|None	None
lower	float|None	None
upper	float|None	None

4.1.629 ForecastPoint

tabelz.analysis.time_series_contracts.ForecastPoint

Field	Type	Default
index	int	required
timestamp	str	required
observed	float|None	None
fitted	float|None	None
forecast	float|None	None
lower	float|None	None
upper	float|None	None
holdout	bool	False

4.1.630 ForecastResult

tabelz.analysis.time_series_contracts.ForecastResult

Field	Type	Default
response_column	str	required
time_column	str|None	required
requested_model	str	required
selected_model	str	required

Field	Type	Default
model_label	str	required
frequency	str	required
sample_count	int	required
training_count	int	required
holdout_count	int	required
dropped_count	int	required
horizon	int	required
confidence_level	float	required
season_length	int|None	required
calendar	Literal['strict','weekdays']	required
deterministic_term	Literal['automatic','none','constant','drift']	required
stationarity_term	Literal['constant','trend']	required
points	tuple[ForecastPoint,...]	required
forecast_rows	tuple[ForecastPoint,...]	required
metrics	ForecastMetrics|None	required
candidates	tuple[ForecastCandidate,...]	required
parameters	tuple[ForecastParameter,...]	required
stationarity	StationarityDiagnostics	required
diagnostics	ResidualDiagnostics	required
interval_method	str	required
converged	bool|None	required
ar_root_min_modulus	float|None	required
ma_root_min_modulus	float|None	required
stationary	bool|None	required
invertible	bool|None	required
warnings	tuple[str,...]	factory:tuple

4.1.631 ResidualDiagnostics

tabelz.analysis.time_series_contracts.ResidualDiagnostics

Field	Type	Default
residual_count	int	required
residual_mean	float	required
residual_std	float	required
acf	tuple[float,...]	required
pacf	tuple[float|None,...]	required
correlation_confidence_level	float	required
correlation_bound	float	required

Field	Type	Default
ljung_box_lag	int|None	required
ljung_box_statistic	float|None	required
ljung_box_p_value	float|None	required
ljung_box_model_df	int	required
ljung_box_results	tuple[tuple[int,float,float|None],...]	required
jarque_bera_statistic	float|None	required
jarque_bera_p_value	float|None	required
arch_lm_lag	int|None	required
arch_lm_statistic	float|None	required
arch_lm_p_value	float|None	required
notes	tuple[str,...]	factory:tuple

4.1.632 SeriesAnalysisConfig

tabelz.analysis.time_series_contracts.SeriesAnalysisConfig

Settings for a standalone time-series diagnostic or transformation preview.

Field	Type	Default
response_column	str	required
time_column	str|None	None
task	Literal['stationarity','random_walk','correlations','difference']	'stationarity'
maximum_lag	int	24
difference_order	int	1
seasonal_difference	bool	False
season_length	int|None	None
missing_policy	Literal['error','drop']	'error'
calendar	Literal['strict','weekdays']	'strict'
stationarity_term	Literal['constant','trend']	'constant'

4.1.633 SeriesAnalysisPoint

tabelz.analysis.time_series_contracts.SeriesAnalysisPoint

Field	Type	Default
index	int	required
timestamp	str	required
original	float|None	required
transformed	float	required

4.1.634 SeriesAnalysisResult

`tabelz.analysis.time_series_contracts.SeriesAnalysisResult`

Field	Type	Default
kind	Literal['stationarity','random_walk', 'correlations','difference']	required
title	str	required
summary	str	required
response_column	str	required
time_column	str|None	required
frequency	str	required
source_count	int	required
sample_count	int	required
dropped_count	int	required
difference_order	int	required
seasonal_difference	bool	required
season_length	int|None	required
recommended_difference_order	int|None	required
differencing_stages	tuple[DifferencingStage,...]	required
points	tuple[SeriesAnalysisPoint,...]	required
stationarity	StationarityDiagnostics	required
correlations	SeriesCorrelationDiagnostics	required
random_walk_assessment	Literal['consistent','not_consistent','inconclusive']	required
warnings	tuple[str,...]	factory:tuple

4.1.635 SeriesCorrelationDiagnostics

`tabelz.analysis.time_series_contracts.SeriesCorrelationDiagnostics`

Field	Type	Default
sample_count	int	required
acf	tuple[float,...]	required
pacf	tuple[float|None,...]	required
correlation_confidence_level	float	required
correlation_bound	float	required
notes	tuple[str,...]	factory:tuple

4.1.636 StationarityDiagnostics

`tabelz.analysis.time_series_contracts.StationarityDiagnostics`

Field	Type	Default
adf_statistic	float|None	required
adf_p_value	float|None	required
adf_lags	int|None	required
kpss_statistic	float|None	required
kpss_p_value	float|None	required
kpss_lags	int|None	required
assessment	Literal['stationary','difference_candidate','inconclusive']	required
plain_language_summary	str	required
differencing_guidance	str	required
deterministic_term	Literal['constant','trend']	'constant'
adf_lag_method	str	'AIC'
kpss_lag_method	str	'automatic'
notes	tuple[str,...]	factory:tuple

4.1.637 InformationAvailabilitySpec

tabelz.analysis.time_series_preparation.InformationAvailabilitySpec

Columns defining observation, publication, and real-time vintage availability.

Field	Type	Default
observation_column	str	required
publication_column	str	required
realtime_start_column	str|None	None
realtime_end_column	str|None	None

4.1.638 MarketCalendarSpec

tabelz.analysis.time_series_preparation.MarketCalendarSpec

Named exchange calendar and explicit frequency conversion settings.

Field	Type	Default
name	str	'NYSE'
timezone	str	'UTC'
frequency	Literal['native','daily','weekly','monthly','quarterly','annual']	'native'
aggregation	Literal['first','last','mean','sum']	'last'

4.1.639 TransformationStep

`tabelz.analysis.time_series_preparation.TransformationStep`

One ordered target transformation with explicit inverse semantics.

Field	Type	Default
`kind`	`Literal['log','difference','simple_return','log_return','growth_rate','detrend','seasonal_adjustment']`	required
`periods`	`int`	1
`annualization_periods`	`int|None`	None

4.1.640 TrendArtifact

`tabelz.analysis.time_series_trend.TrendArtifact`

What reproduces the fitted trend at a row's time position.

Field	Type	Default
`response`	`str`	required
`model`	`Literal['linear','quadratic','exponential','exponential_nls','scurve']`	required
`parameters`	`tuple[float,...]`	required
`row_ids`	`tuple[str,...]`	required

4.1.641 TrendConfig

`tabelz.analysis.time_series_trend.TrendConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`response`	`str`	required
`order_column`	`str|None`	None
`model`	`Literal['linear','quadratic','exponential','exponential_nls','scurve']`	'linear'
`horizon`	`int`	0

horizon
Ge(ge=0); Le(le=1000)

4.1.642 TrendParameter

`tabelz.analysis.time_series_trend.TrendParameter`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
estimate	float	required
standard_error	float|None	None

4.1.643 TrendResult

tabelz.analysis.time_series_trend.TrendResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	TrendConfig	required
equation	str	required
parameterization	str	required
parameters	tuple[TrendParameter,...]	required
pearl_reed	tuple[TrendParameter,...]	()
starting_values	tuple[float,...]	()
fitted_count	int	required
series_length	int	required
missing_count	int	required
excluded_count	int	required
mae	float	required
mse	float	required
rmse	float	required
mape	float|None	required
mape_note	str	''
fitted	tuple[tuple[float,float,float],...]	required
extrapolation	tuple[tuple[float,float],...]	()
extrapolation_note	str	''
artifact	TrendArtifact	required
warnings	tuple[str,...]	()

4.1.644 RollingValidationConfig

tabelz.analysis.time_series_validation.RollingValidationConfig

Rolling-origin design, including a final period excluded from selection.

Field	Type	Default
initial_window	int	required
horizon	int	required
step	int	1

Field	Type	Default
window	Literal['expanding','sliding']	'expanding'
sliding_window	int|None	None
origins	tuple[int,...]	()
final_evaluation	int	0
selection_holdout	int	0

4.1.645 DerivedFeatureSpec

tabelz.analysis.time_series_workbench_contracts.DerivedFeatureSpec

One predictor feature created without changing the source dataset.

Field	Type	Default
column	str	required
kind	Literal['level','lag','rolling_mean', 'rolling_std']	'level'
periods	int	1
window	int	3

4.1.646 DynamicRegressionResult

tabelz.analysis.time_series_workbench_contracts.DynamicRegressionResult

Field	Type	Default
model_label	str	required
response_column	str	required
feature_names	tuple[str,...]	required
forecast_rows	tuple[ForecastPoint,...]	required
parameters	tuple[ForecastParameter,...]	required
converged	bool	required
conditional_on_predictors	bool	required
predictor_uncertainty_included	bool	required
scenario_summary	tuple[dict[str,object],...]	required
time_column	str|None	required
sample_count	int	required
confidence_level	float	required
points	tuple[ForecastPoint,...]	required
specification	dict[str,object]	required
specification_fingerprint	str	required

Field	Type	Default
data_fingerprint	str	required
warnings	tuple[str,...]	()

4.1.647 DynamicRegressionSpec

tabelz.analysis.time_series_workbench_contracts.DynamicRegressionSpec

SARIMAX/distributed-lag model with explicit future predictor scenarios.

Field	Type	Default
response_column	str	required
features	tuple[DerivedFeatureSpec,...]	required
future_scenarios	tuple[FuturePredictorScenario,...]	required
time_column	str|None	None
arima_order	tuple[int,int,int]	(1,0,0)
seasonal_order	tuple[int,int,int,int]	(0,0,0,0)
deterministic_term	Literal['none','constant','trend']	'constant'
confidence_level	float	0.95
predictor_draws	int	0
random_seed	int	0
future_time_labels	tuple[str,...]	()

4.1.648 DynamicRegressionStudyResult

tabelz.analysis.time_series_workbench_contracts.DynamicRegressionStudyResult

Field	Type	Default
study_name	str	required
specification	dict[str,object]	required
specification_fingerprint	str	required
data_fingerprint	str	required
origins	tuple[int,...]	required
rolling_forecasts	tuple[OriginForecast,...]	required
accuracy_by_horizon	tuple[HorizonAccuracy,...]	required
overall_metrics	ForecastMetrics	required
naive_benchmark_metrics	ForecastMetrics	required
final_evaluation_metrics	ForecastMetrics|None	required
final_evaluation_forecasts	tuple[ForecastPoint,...]	required

Field	Type	Default
future_forecast	DynamicRegressionResult	required
original_sample_count	int	required
effective_sample_count	int	required
future_training_count	int	required
information_policy	str	required
warnings	tuple[str,...]	()

4.1.649 DynamicRegressionStudySpec

tabelz.analysis.time_series_workbench_contracts.DynamicRegressionStudySpec

Replayable rolling study for a dynamic regression specification.

Field	Type	Default
regression	DynamicRegressionSpec	required
validation	RollingValidationConfig	required
availability	InformationAvailabilitySpec|None	None
historical_predictor_policy	Literal['available','realized']	'available'
study_name	str	'Dynamic-regressionstudy'
schema_version	str	'1.0.0'

4.1.650 ForecastStudyResult

tabelz.analysis.time_series_workbench_contracts.ForecastStudyResult

Field	Type	Default
study_name	str	required
specification	dict[str,object]	required
specification_fingerprint	str	required
data_fingerprint	str	required
origins	tuple[int,...]	required
rolling_forecasts	tuple[OriginForecast,...]	required
accuracy_by_horizon	tuple[HorizonAccuracy,...]	required
overall_metrics	ForecastMetrics|None	required
naive_benchmark_metrics	ForecastMetrics	required

Field	Type	Default
seasonal_naive_benchmark_metrics	ForecastMetrics|None	required
final_evaluation_metrics	ForecastMetrics|None	required
final_evaluation_forecasts	tuple[ForecastPoint,...]	required
future_forecast	ForecastResult	required
original_sample_count	int	required
effective_sample_count	int	required
future_training_count	int	required
transformation_history	tuple[dict[str,object],...]	required
information_policy	str	required
calendar_alignment	dict[str,object]|None	None
warnings	tuple[str,...]	factory:tuple
reconstruction	dict[str,object]|None	None
evaluation	dict[str,object]|None	None

4.1.651 ForecastStudySpec

tabelz.analysis.time_series_workbench_contracts.ForecastStudySpec

Complete replayable specification for one forecast study.

Field	Type	Default
forecast	ForecastConfig	required
validation	RollingValidationConfig	required
transformations	tuple[TransformationStep,...]	()
back_transform_center	Literal['median','mean']	'median'
calendar	MarketCalendarSpec|None	None
availability	InformationAvailabilitySpec|None	None
predictors	tuple[DerivedFeatureSpec,...]	()
response_unit	str|None	None
frequency	Literal['native','daily','weekly','monthly','quarterly','annual']	'native'
aggregation	Literal['first','last','mean','sum']	'last'
timezone	str	'UTC'
study_name	str	'Forecaststudy'
schema_version	str	'1.0.0'

Field	Type	Default
evaluation_convention	Literal['legacy','corrected-2026-09']	'corrected-2026-09'

4.1.652 FuturePredictorScenario

tabelz.analysis.time_series_workbench_contracts.FuturePredictorScenario

Explicit future path for one predictor, optionally with standard errors.

Field	Type	Default
column	str	required
values	tuple[float,...]	required
standard_errors	tuple[float,...]	()

4.1.653 HorizonAccuracy

tabelz.analysis.time_series_workbench_contracts.HorizonAccuracy

Field	Type	Default
horizon	int	required
count	int	required
metrics	ForecastMetrics	required
naive_benchmark_metrics	ForecastMetrics	required
seasonal_naive_benchmark_metrics	ForecastMetrics|None	None

4.1.654 OriginForecast

tabelz.analysis.time_series_workbench_contracts.OriginForecast

Field	Type	Default
origin	int	required
origin_label	str	required
training_start	int	required
training_count	int	required
horizon	int	required
target_label	str	required
actual	float	required
forecast	float	required
lower	float|None	required
upper	float|None	required

Field	Type	Default
model	str	required

4.1.655 FirstStage

tabelz.analysis.two_stage_least_squares.FirstStage

Common strict, immutable model baseline for public contracts.

Field	Type	Default
endogenous	str	required
r_squared	float	required
partial_r_squared	float	required
f_value	float|None	required
df1	int	required
df2	int	required
p_value	float|None	required
robust_f	float|None	None
robust_p	float|None	None

4.1.656 IvCoefficient

tabelz.analysis.two_stage_least_squares.IvCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
role	Literal['constant','exogenous','endogenous']	required
estimate	float	required
standard_error	float	required
t_value	float|None	required
p_value	float|None	required
lower	float	required
upper	float	required

4.1.657 SarganTest

tabelz.analysis.two_stage_least_squares.SarganTest

Common strict, immutable model baseline for public contracts.

Field	Type	Default
statistic	float	required
df	int	required
p_value	float	required

4.1.658 TwoStageConfig

tabelz.analysis.two_stage_least_squares.TwoStageConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
endogenous	tuple[str,...]	required
exogenous	tuple[str,...]	()
instruments	tuple[str,...]	required
covariance	Literal['classical','hc1']	'classical'
confidence_level	float	0.95

endogenous
MinLen(min_length=1); MaxLen(max_length=10)

exogenous
MaxLen(max_length=40)

instruments
MinLen(min_length=1); MaxLen(max_length=40)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.659 TwoStageResult

tabelz.analysis.two_stage_least_squares.TwoStageResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	TwoStageConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
coefficients	tuple[IvCoefficient,...]	required
first_stages	tuple[FirstStage,...]	required
sargan	SarganTest|None	None
residual_standard_error	float	required
r_squared	float	required

Field	Type	Default
wald_f	float|None	required
wald_df1	int	required
wald_df2	int	required
wald_p	float|None	required
notes	tuple[str,...]	()

4.1.660 VariablesAcceptanceConfig

tabelz.analysis.variables_acceptance.VariablesAcceptanceConfig

Strict request for one one-sided normal-process variables plan.

Field	Type	Default
spec_side	Literal['upper','lower']	required
spec_limit	float	required
sigma_status	Literal['known','unknown']	required
sigma	float|None	None
aql	float	required
ltpd	float	required
producer_risk	float	0.05
consumer_risk	float	0.1
max_sample_size	int	5000
oc_points	int	101

aql
Gt(gt=0.0); Lt(lt=1.0)
ltpd
Gt(gt=0.0); Lt(lt=1.0)
producer_risk
Gt(gt=0.0); Lt(lt=1.0)
consumer_risk
Gt(gt=0.0); Lt(lt=1.0)
max_sample_size
Ge(ge=2); Le(le=5000)
oc_points
Ge(ge=2); Le(le=10001)

4.1.661 VariablesAcceptanceResult

tabelz.analysis.variables_acceptance.VariablesAcceptanceResult

One planned variables acceptance plan with its operating characteristic.

Field	Type	Default
method	str	required
distribution	Literal['normal','noncentral_t']	required
sigma_status	Literal['known','unknown']	required
spec_side	Literal['upper','lower']	required
spec_limit	float	required
sigma	float|None	required
aql	float	required
ltpd	float	required
producer_risk	float	required
consumer_risk	float	required
sample_size	int	required
acceptance_constant	float	required
probability_accept_aql	float	required
probability_accept_ltpd	float	required
oc_points	tuple[dict[str,float],...]	()
instruction	str	required
assumptions	tuple[str,...]	()
warnings	tuple[str,...]	()
implementation_version	str	'1.0.0'

4.1.662 MultivariateTimeSeriesResult

tabelz.analysis.vector_autoregression.MultivariateTimeSeriesResult

Field	Type	Default
columns	tuple[str,...]	required
model_label	str	required
sample_count	int	required
lag_order	int	required
cointegration_rank	int	required
johansen_trace	tuple[dict[str,float|int|bool],...]	required
forecasts	tuple[dict[str,object],...]	required
granger_predictability	tuple[dict[str,object],...]	required
impulse_responses	tuple[dict[str,object],...]	required
diagnostics	dict[str,object]	required
specification	dict[str,object]	required
specification_fingerprint	str	required

Field	Type	Default
`data_fingerprint`	`str`	`required`
`reproducibility`	`dict[str,str]`	`required`
`warnings`	`tuple[str,...]`	`()`

4.1.663 WarrantyCohort

`tabelz.analysis.warranty.WarrantyCohort`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`label`	`str`	`required`
`shipment_period`	`int`	`required`
`shipped`	`int`	`required`
`failures`	`int`	`required`
`survivors`	`int`	`required`
`age`	`float`	`required`
`excluded_failures`	`int`	`required`

shipment_period
Ge(ge=1)
shipped
Ge(ge=0)
failures
Ge(ge=0)
survivors
Ge(ge=0)
age
Ge(ge=0.0)
excluded_failures
Ge(ge=0)

4.1.664 WarrantyConversion

`tabelz.analysis.warranty.WarrantyConversion`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`return_axis`	`Literal['calendar','age']`	`required`
`period_width`	`float`	`required`
`period_unit`	`str`	`required`
`last_period`	`Literal['incomplete','complete']`	`required`
`as_of_period`	`int`	`required`

Field	Type	Default
shipped_total	int	required
failures_used	int	required
survivors_total	int	required
excluded_failures	int	required
excluded_shipments	int	required
cohorts	tuple[WarrantyCohort,...]	required
records	tuple[WarrantyRecord,...]	required
assumptions	tuple[str,...]	required
source_count	int	required
excluded_row_count	int	required

as_of_period
Ge(ge=1)
shipped_total
Ge(ge=0)
failures_used
Ge(ge=0)
survivors_total
Ge(ge=0)
excluded_failures
Ge(ge=0)
excluded_shipments
Ge(ge=0)
source_count
Ge(ge=0)
excluded_row_count
Ge(ge=0)

4.1.665 WarrantyPreprocessConfig

`tabelz.analysis.warranty.WarrantyPreprocessConfig`

Pre-process settings: the triangle only.

Field	Type	Default
shipped	str	required
returns	tuple[str,...]	required
return_axis	Literal['calendar','age']	'calendar'
period_width	float	1.0
period_unit	str	'month'
last_period	Literal['incomplete','complete']	'incomplete'
shipment_label	str|None	None

shipped
MinLen(min_length=1)

period_width
Gt(gt=0.0)

period_unit
MinLen(min_length=1); MaxLen(max_length=24)

4.1.666 WarrantyPreprocessResult

`tabelz.analysis.warranty.WarrantyPreprocessResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`conversion`	`WarrantyConversion`	`required`
`warnings`	`tuple[str,...]`	`()`

4.1.667 WarrantyRecord

`tabelz.analysis.warranty.WarrantyRecord`

One censored record with its frequency and cohort.

Field	Type	Default
`cohort`	`str`	`required`
`start`	`float|None`	`required`
`end`	`float|None`	`required`
`frequency`	`int`	`required`
`kind`	`Literal['interval','right']`	`required`

frequency
Ge(ge=0)

4.1.668 CohortPrediction

`tabelz.analysis.warranty_prediction.CohortPrediction`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`label`	`str`	`required`
`survivors`	`float`	`required`
`age`	`float`	`required`
`eligible`	`bool`	`required`
`expected`	`float|None`	`required`
`note`	`str`	`''`

survivors
Ge(ge=0.0)
age
Ge(ge=0.0)

4.1.669 FutureShipmentPrediction

`tabelz.analysis.warranty_prediction.FutureShipmentPrediction`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`units`	`float`	`required`
`time`	`float`	`required`
`expected`	`float`	`required`

units
Ge(ge=0.0)
time
Ge(ge=0.0)
expected
Ge(ge=0.0)

4.1.670 PeriodPrediction

`tabelz.analysis.warranty_prediction.PeriodPrediction`

Cumulative expectation through “horizon“; “lower“/“upper“ bound the cumulative expectation, not the increment. “None“ where a contribution is undefined.

Field	Type	Default
`horizon`	`float`	`required`
`expected`	`float|None`	`required`
`increment`	`float|None`	`required`
`lower`	`float|None`	`None`
`upper`	`float|None`	`None`

4.1.671 WarrantyPredictionConfig

`tabelz.analysis.warranty_prediction.WarrantyPredictionConfig`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`source`	`Literal['triangle','censoring_table']`	`'triangle'`
`shipped`	`str|None`	`None`
`returns`	`tuple[str,...]`	`()`

Field	Type	Default
return_axis	Literal['calendar','age']	'calendar'
period_width	float	1.0
period_unit	str	'month'
last_period	Literal['incomplete','complete']	'incomplete'
start	str|None	None
end	str|None	None
frequency	str|None	None
family	Literal['exponential','weibull','log normal','loglogistic']	'weibull'
horizon	float	12.0
warranty_limit	float|None	None
future_shipments	tuple[str,...]	()
unit_cost	float|None	None
confidence_level	float	0.95

period_width
Gt(gt=0.0)

period_unit
MinLen(min_length=1); MaxLen(max_length=24)

horizon
Ge(ge=0.0)

warranty_limit
Gt(gt=0.0)

unit_cost
Ge(ge=0.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.672 WarrantyPredictionResult

`tabelz.analysis.warranty_prediction.WarrantyPredictionResult`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
family	Literal['exponential','weibull','log normal','loglogistic']	required
horizon	float	required
warranty_limit	float|None	required
confidence_level	float	required
period_unit	str	required
conversion	WarrantyConversion|None	required
parameters	tuple[LifeParameter,...]	required

Field	Type	Default
log_likelihood	float	required
cohorts	tuple[CohortPrediction, ...]	required
future	tuple[FutureShipmentPrediction, ...]	required
field_expected	float|None	required
future_expected	float	required
expected	float|None	required
eligible_units	float	required
partial_expected	float|None	None
omitted_units	float	0.0
omitted_cohorts	tuple[str, ...]	()
lower	float|None	None
upper	float|None	None
interval_status	str	required
unit_cost	float|None	None
expected_cost	float|None	None
cost_lower	float|None	None
cost_upper	float|None	None
periods	tuple[PeriodPrediction, ...]	required
warnings	tuple[str, ...]	()

field_expected
Ge(ge=0.0)
future_expected
Ge(ge=0.0)
expected
Ge(ge=0.0)
eligible_units
Ge(ge=0.0)
partial_expected
Ge(ge=0.0)
omitted_units
Ge(ge=0.0)

4.1.673 PowerLikelihood

`tabelz.analysis.weight_estimation.PowerLikelihood`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
power	float	required
log_likelihood	float	required

Field	Type	Default
weighted_sse	float	required

4.1.674 WeightEstimationConfig

tabelz.analysis.weight_estimation.WeightEstimationConfig

Common strict, immutable model baseline for public contracts.

Field	Type	Default
response	str	required
predictors	tuple[str, ...]	required
weight_variable	str	required
power_from	float	-2.0
power_to	float	2.0
power_step	float	0.5
confidence_level	float	0.95

predictors
MinLen(min_length=1); MaxLen(max_length=64)

power_step
Gt(gt=0.0)

confidence_level
Gt(gt=0.5); Lt(lt=1.0)

4.1.675 WeightEstimationResult

tabelz.analysis.weight_estimation.WeightEstimationResult

Common strict, immutable model baseline for public contracts.

Field	Type	Default
config	WeightEstimationConfig	required
n	int	required
missing_count	int	required
excluded_count	int	required
grid	tuple[PowerLikelihood, ...]	required
selected_power	float	required
at_boundary	bool	required
coefficients	tuple[WlsCoefficient, ...]	required
r_squared	float	required
adjusted_r_squared	float	required
standard_error	float	required
f_value	float|None	required

Field	Type	Default
df1	int	required
df2	int	required
p_value	float|None	required
notes	tuple[str,...]	()
warnings	tuple[str,...]	()

4.1.676 WlsCoefficient

tabelz.analysis.weight_estimation.WlsCoefficient

Common strict, immutable model baseline for public contracts.

Field	Type	Default
term	str	required
estimate	float	required
standard_error	float	required
t_value	float|None	required
p_value	float|None	required
lower	float	required
upper	float	required

4.2 tabelz.contracts

4.2.1 ColumnMetadata

tabelz.contracts.column_metadata.ColumnMetadata

Common strict, immutable model baseline for public contracts.

Field	Type	Default
variable_label	str	''
value_labels	tuple[ValueLabel,...]	()
measurement_level	Literal['nominal','ordinal','scale'] |None	None
display	DisplayFormat|None	None
user_missing	UserMissing|None	None

variable_label
MaxLen(max_length=256)

value_labels
MaxLen(max_length=5000)

4.2.2 DisplayFormat

tabelz.contracts.column_metadata.DisplayFormat

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`decimals`	`int|None`	`None`
`width`	`int|None`	`None`
`date_format`	`str|None`	`None`

decimals
Ge(ge=0); Le(le=16)
width
Ge(ge=1); Le(le=64)
date_format
MinLen(min_length=1); MaxLen(max_length=32)

4.2.3 TableMetadata

`tabelz.contracts.column_metadata.TableMetadata`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`columns`	`dict[str,ColumnMetadata]`	`factory:dict`

4.2.4 UserMissing

`tabelz.contracts.column_metadata.UserMissing`

Up to three discrete values, or one closed range plus at most one discrete value.

Field	Type	Default
`values`	`tuple[bool|int|float|str,...]`	`()`
`low`	`float|None`	`None`
`high`	`float|None`	`None`

4.2.5 ValueLabel

`tabelz.contracts.column_metadata.ValueLabel`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`value`	`bool|int|float|str`	`required`
`label`	`str`	`required`

label
MinLen(min_length=1); MaxLen(max_length=256)

4.2.6 ImportedFileMetadata

`tabelz.contracts.imported_file.ImportedFileMetadata`

The original metadata of one imported file, keyed by current column name.

Field	Type	Default
`source`	`ImportedFileSource`	`required`
`columns`	`dict[str,OriginalColumn]`	`factory:dict`
`label_sets`	`tuple[OriginalLabelSet,...]`	`()`
`special_missing`	`tuple[SpecialMissingColumn,...]`	`()`

4.2.7 ImportedFileSource

`tabelz.contracts.imported_file.ImportedFileSource`

Which file the metadata came from, and how its text was decoded.

Field	Type	Default
`original_name`	`str`	`required`
`sha256`	`str`	`required`
`package`	`Literal['SPSS','Stata','SAS','R']`	`required`
`file_format`	`Literal['sav','zsav','por','dta','sas 7bdat','xpt','rds','rdata']`	`required`
`format_version`	`str`	`''`
`encoding`	`str`	`''`
`encoding_declared`	`bool`	`False`
`library`	`str`	`''`
`catalog`	`str`	`''`
`catalog_sha256`	`str`	`''`

4.2.8 MissingCode

`tabelz.contracts.imported_file.MissingCode`

One special missing-value code present in a column, or an R tag its labels name.

Field	Type	Default
`code`	`str`	`required`
`label`	`str`	`''`
`count`	`int`	`required`

code
MinLen(min_length=2)

count
Ge(ge=0)

4.2.9 OriginalColumn

`tabelz.contracts.imported_file.OriginalColumn`

One variable's declared storage and verbatim display format.

Field	Type	Default
`source_column`	`str`	`required`
`storage_type`	`str`	`required`
`format`	`str`	`''`
`storage_width`	`int|None`	`None`
`display_width`	`int|None`	`None`
`label_set`	`str`	`''`
`format_applies`	`bool`	`True`
`labels_apply`	`bool`	`True`
`changes`	`tuple[str,...]`	`()`

source_column
MinLen(min_length=1)

4.2.10 OriginalLabel

`tabelz.contracts.imported_file.OriginalLabel`

One code of a value-label set, typed as the file declares it, and its label.

Field	Type	Default
`value`	`bool|int|float|str`	`required`
`label`	`str`	`required`

4.2.11 OriginalLabelSet

`tabelz.contracts.imported_file.OriginalLabelSet`

A named value-label set as the file (or its SAS format catalog) defines it.

Field	Type	Default
`name`	`str`	`required`
`labels`	`tuple[OriginalLabel,...]`	`()`
`columns`	`tuple[str,...]`	`()`

name
MinLen(min_length=1)

4.2.12 SpecialMissingColumn

`tabelz.contracts.imported_file.SpecialMissingColumn`

The special missing-value codes of one measurement column.

Field	Type	Default
column	str	required
code_column	str	''
codes	tuple[MissingCode, ...]	required
discarded	bool	False
changes	tuple[str, ...]	()

column
MinLen(min_length=1)

codes
MinLen(min_length=1)

4.2.13 ResponseSet

tabelz.contracts.multiple_response.ResponseSet

Common strict, immutable model baseline for public contracts.

Field	Type	Default
name	str	required
kind	Literal['dichotomy', 'category']	required
columns	tuple[str, ...]	required
counted_value	bool|int|float|str|None	None
label	str	''

name
MinLen(min_length=1); MaxLen(max_length=64)

columns
MinLen(min_length=2); MaxLen(max_length=50)

label
MaxLen(max_length=256)

4.2.14 EngineVersion

tabelz.contracts.operations.EngineVersion

Version capture for the engine responsible for a contract payload.

Field	Type	Default
engine	str	required
version	str	required
python_version	str|None	None
adapter	str|None	None

engine
MinLen(min_length=1)

version
MinLen(min_length=1)

4.2.15 OperationSpec

`tabelz.contracts.operations.OperationSpec`

Immutable, serializable description of a project state transition.

Field	Type	Default
operation_id	str	required
operation_type	str	required
schema_version	str	required
implementation_version	str	required
parameters	JsonObject	factory:dict
input_tables	tuple[TableRef,...]	factory:tuple
input_results	tuple[ResultRef,...]	factory:tuple
input_populations	tuple[PopulationRef,...]	factory:tuple
output_tables	tuple[TableRef,...]	factory:tuple
output_results	tuple[ResultRef,...]	factory:tuple
output_populations	tuple[PopulationRef,...]	factory:tuple
random_seed	int|None	None
created_at	pydantic.types.AwareDatetime	required
actor_type	ActorType	required
label	str|None	None
engine_version	EngineVersion|None	None

operation_id
MinLen(min_length=36); MaxLen(max_length=36)
operation_type
MinLen(min_length=1)
schema_version
MinLen(min_length=1)
implementation_version
MinLen(min_length=1)

4.2.16 PopulationRef

`tabelz.contracts.references.PopulationRef`

Stable reference to a population defined over a table.

Field	Type	Default
population_id	str	required
table_id	str	required

Field	Type	Default
population_name	str|None	None

population_id
MinLen(min_length=36); MaxLen(max_length=36)
table_id
MinLen(min_length=36); MaxLen(max_length=36)
population_name
MinLen(min_length=1)

4.2.17 ResultRef

tabelz.contracts.references.ResultRef

Stable reference to a typed operation result.

Field	Type	Default
result_id	str	required
result_type	str	required

result_id
MinLen(min_length=36); MaxLen(max_length=36)
result_type
MinLen(min_length=1)

4.2.18 TableRef

tabelz.contracts.references.TableRef

Stable reference to a logical table.

Field	Type	Default
table_id	str	required
table_name	str|None	None

table_id
MinLen(min_length=36); MaxLen(max_length=36)
table_name
MinLen(min_length=1)

4.2.19 ArtifactRef

tabelz.contracts.reproduction.ArtifactRef

A fitted artifact (a model, basis or clustering) retained for reuse.

Field	Type	Default
artifact_id	str	required
artifact_type	str	required

artifact_id
MinLen(min_length=36); MaxLen(max_length=36)

artifact_type
MinLen(min_length=1)

4.2.20 BoundedSpecRef

tabelz.contracts.reproduction.BoundedSpecRef

A bounded plot or report specification stored with the results.

Field	Type	Default
spec_id	str	required
spec_type	str	required

spec_id
MinLen(min_length=36); MaxLen(max_length=36)

spec_type
MinLen(min_length=1)

4.2.21 MissingValueEncoding

tabelz.contracts.reproduction.MissingValueEncoding

Import-time missing-value handling, so a replay reads the same nulls.

Field	Type	Default
codes	tuple[str,...]	()
empty_string_is_null	bool	True

4.2.22 NamedOutput

tabelz.contracts.reproduction.NamedOutput

One output actually returned, carrying exactly the reference its kind needs.

Field	Type	Default
name	str	required
kind	Literal['table','analysis_result','fitted_artifact','plot_spec','report_spec']	required
table	TableStateRef|None	None

Field	Type	Default
result	ResultRef|None	None
artifact	ArtifactRef|None	None
spec	BoundedSpecRef|None	None

4.2.23 OperationOutcome

tabelz.contracts.reproduction.OperationOutcome

What one executed operation returned.

Field	Type	Default
operation_id	str	required
outputs	tuple[NamedOutput,...]	()

operation_id
MinLen(min_length=36); MaxLen(max_length=36)

4.2.24 OperationOutputContract

tabelz.contracts.reproduction.OperationOutputContract

Every output a registered operation may return.

Field	Type	Default
declarations	tuple[OutputDeclaration,...]	required

declarations
MinLen(min_length=1)

4.2.25 OutputDeclaration

tabelz.contracts.reproduction.OutputDeclaration

One output an operation may return, by stable name and kind.

Field	Type	Default
name	str	required
kind	Literal['table','analysis_result','fitted_artifact','plot_spec','report_spec']	required
required	bool	True

4.2.26 RecipeTarget

tabelz.contracts.reproduction.RecipeTarget

A final table or retained result a recipe can rebuild, by stable ID.

Field	Type	Default
target_id	str	required
kind	Literal['table','result']	required
label	str	required
table	TableStateRef|None	None
result	ResultRef|None	None
producer_operation_id	str|None	None
dependencies	tuple[str,...]	()

label
MinLen(min_length=1)

4.2.27 RecipeTargetSet

tabelz.contracts.reproduction.RecipeTargetSet

Every named target of a recipe, with a closed and acyclic dependency graph.

Field	Type	Default
targets	tuple[RecipeTarget,...]	()

4.2.28 RowIdentityEncoding

tabelz.contracts.reproduction.RowIdentityEncoding

How the snapshot's stable row identities are stored.

Field	Type	Default
encoding	Literal['member','column']	required
member	str|None	None
column	str|None	None

column
MinLen(min_length=1)

4.2.29 SnapshotColumn

tabelz.contracts.reproduction.SnapshotColumn

One column's logical type, as imported, before any later edit.

Field	Type	Default
name	str	required

Field	Type	Default
logical_type	Literal['string','int64','float64','boolean','date','datetime','time','duration','categorical','binary','null']	required
physical_dtype	str	required
nullable	bool	True
time_zone	str|None	None
datetime_unit	Literal['ms','us','ns']|None	None
categories	tuple[str,...]|None	None

name
MinLen(min_length=1)
physical_dtype
MinLen(min_length=1)
time_zone
MinLen(min_length=1)

4.2.30 SourceProvenance

tabelz.contracts.reproduction.SourceProvenance

Where a snapshot came from. Never credentials or connection secrets.

Field	Type	Default
kind	Literal['csv','excel','sqlite','duckdb','postgresql','sqlserver','spss','stata','sas','parquet','arrow','r','matlab','hdf5','clipboard','generated']	required
original_name	str|None	None
source_checksum	str|None	None
sheet	str|None	None
cell_range	str|None	None
query	str|None	None
query_parameters	JsonObject	factory:dict
captured_at	pydantic.types.AwareDatetime	required

original_name
MinLen(min_length=1)
sheet
MinLen(min_length=1)
cell_range
MinLen(min_length=1)
query
MinLen(min_length=1)

4.2.31 SourceSnapshotMetadata

`tabelz.contracts.reproduction.SourceSnapshotMetadata`

An embedded starting table and everything needed to read it back exactly.

Field	Type	Default
snapshot_id	str	required
table_id	str	required
revision	int	required
member	str	required
format	Literal['csv','parquet','arrow_ipc']	required
columns	tuple[SnapshotColumn,...]	required
row_identity	RowIdentityEncoding	required
row_count	int	required
checksum	str	required
size	int	required
row_identity_checksum	str|None	None
row_identity_size	int|None	None
logical_fingerprint	str	required
missing_values	MissingValueEncoding	factory:MissingValueEncoding
import_settings	JsonObject	factory:dict
provenance	SourceProvenance	required

snapshot_id
MinLen(min_length=36); MaxLen(max_length=36)

table_id
MinLen(min_length=36); MaxLen(max_length=36)

revision
Ge(ge=0)

row_count
Ge(ge=0)

size
Ge(ge=0)

row_identity_size
Ge(ge=0)

4.2.32 TableStateRef

`tabelz.contracts.reproduction.TableStateRef`

One state of one tab: its table ID at one data revision, from one origin.

Field	Type	Default
table_id	str	required
revision	int	required
origin	Literal['snapshot','operation']	required
snapshot_id	str|None	None
operation_id	str|None	None

table_id
MinLen(min_length=36); MaxLen(max_length=36)

revision
Ge(ge=0)

4.2.33 PivotSpec

tabelz.contracts.reshape.PivotSpec

Strict specification for one value-family long-to-wide operation.

Field	Type	Default
identifiers	tuple[str,...]	required
measurement_name_column	str	required
value_column	str	required
measurement_names	tuple[str,...]	required
null_identifier_policy	Literal['group']	'group'

identifiers
MinLen(min_length=1); MaxLen(max_length=127)

measurement_name_column
MinLen(min_length=1); MaxLen(max_length=120)

value_column
MinLen(min_length=1); MaxLen(max_length=120)

measurement_names
MinLen(min_length=1); MaxLen(max_length=127)

4.2.34 ReshapeLimits

tabelz.contracts.reshape.ReshapeLimits

Explicit bounds checked before a reshape output is materialized.

Field	Type	Default
max_input_rows	int	1000000
max_output_rows	int	2000000
max_output_cells	int	8000000

Field	Type	Default
max_output_width	int	128
preview_rows	int	100

max_input_rows
Ge(ge=0); Le(le=10000000)
max_output_rows
Ge(ge=0); Le(le=10000000)
max_output_cells
Ge(ge=0); Le(le=50000000)
max_output_width
Ge(ge=1); Le(le=128)
preview_rows
Ge(ge=1); Le(le=500)

4.2.35 StackSpec

tabelz.contracts.reshape.StackSpec

Strict specification for one value-family wide-to-long operation.

Field	Type	Default
identifiers	tuple[str,...]	required
measurements	tuple[str,...]	required
measurement_name_column	str	'measurement'
value_column	str	'value'
null_identifier_policy	Literal['group']	'group'

identifiers
MinLen(min_length=1); MaxLen(max_length=127)
measurements
MinLen(min_length=2); MaxLen(max_length=127)
measurement_name_column
MinLen(min_length=1); MaxLen(max_length=120)
value_column
MinLen(min_length=1); MaxLen(max_length=120)

4.2.36 StratumCombination

tabelz.contracts.survey_design.StratumCombination

One original first-stage stratum merged into a declared combined stratum.

Field	Type	Default
stratum	str	required
combined_stratum	str	required

stratum
MinLen(min_length=1); MaxLen(max_length=256)
combined_stratum
MinLen(min_length=1); MaxLen(max_length=256)

4.2.37 SurveyCalibration

tabelz.contracts.survey_design.SurveyCalibration

Recorded one-step known-total calibration, with original design and variance inputs.

Field	Type	Default
original_design	SurveyDesign	required
method	Literal['poststratification','raking','linear']	required
targets	tuple[SurveyCalibrationTarget,...]	required
lower_bound	float|None	None
upper_bound	float|None	None
tolerance	float	required
max_iterations	int	required
source_row_ids	tuple[str,...]	required
source_fingerprint	str	required
output_fingerprint	str	required
original_design_df	int	required
implementation_version	Literal['1.0.0']	'1.0.0'

targets
MinLen(min_length=1); MaxLen(max_length=50)
lower_bound
Gt(gt=0)
upper_bound
Gt(gt=0)
tolerance
Gt(gt=0); Le(le=0.01)
max_iterations
Ge(ge=1); Le(le=1000)
source_row_ids
MinLen(min_length=2); MaxLen(max_length=20000)
original_design_df
Gt(gt=0)

4.2.38 SurveyCalibrationTarget

`tabelz.contracts.survey_design.SurveyCalibrationTarget`

Known total of a numeric auxiliary, category indicator, or intercept (column=None).

Field	Type	Default
`column`	`str|None`	`None`
`level`	`str|None`	`None`
`total`	`float`	`required`

column
MinLen(min_length=1)

4.2.39 SurveyDesign

`tabelz.contracts.survey_design.SurveyDesign`

A table's complex-samples design.

Field	Type	Default
`weight`	`str`	`required`
`calibration`	`SurveyCalibration|None`	`None`
`variance`	`Literal['linearization','replicate']`	`'linearization'`
`stage1`	`SurveyStage`	`SurveyStageinstance`
`stage2`	`SurveyStage|None`	`None`
`nested_units`	`bool`	`False`
`single_psu`	`Literal['refuse','centered']`	`'refuse'`
`certainty_strata`	`tuple[str,...]`	`()`
`combine_strata`	`tuple[StratumCombination,...]`	`()`
`combination_justification`	`str`	`''`
`replicates`	`SurveyReplicates|None`	`None`

weight
MinLen(min_length=1)
certainty_strata
MaxLen(max_length=1000)
combine_strata
MaxLen(max_length=1000)
combination_justification
MaxLen(max_length=2000)

4.2.40 SurveyReplicateGeneration

`tabelz.contracts.survey_design.SurveyReplicateGeneration`

Bounded immutable generation provenance; variance conventions live in SurveyReplicates.

Field	Type	Default
algorithm	Literal['jk1','jkn','brr','fay','rao_wu']	required
source_design	SurveyDesign	required
source_row_ids	tuple[str,...]	required
psu_for_row	tuple[int,...]	required
psu_labels	tuple[tuple[str,str],...]	required
psu_strata	tuple[int,...]	required
source_fingerprint	str	required
output_fingerprint	str	required
seed	int|None	None
hadamard_columns	tuple[int,...]	()
sampling_assumption	Literal['first_stage_srs_psu']	'first_stage_srs_psu'
implementation_version	Literal['1.0.0']	'1.0.0'

source_row_ids
MinLen(min_length=2); MaxLen(max_length=20000)

psu_for_row
MinLen(min_length=2); MaxLen(max_length=20000)

psu_labels
MinLen(min_length=2); MaxLen(max_length=20000)

psu_strata
MinLen(min_length=2); MaxLen(max_length=20000)

seed
Ge(ge=0); Le(le=4294967295)

4.2.41 SurveyReplicates

`tabelz.contracts.survey_design.SurveyReplicates`

Supplied replicate weights and every convention needed to use them.

Field	Type	Default
columns	tuple[str,...]	required
method	Literal['brr','fay','jk1','jkn','bootstrap','sdr','other']	required
weight_type	Literal['full','multiplier']	required
scale	float	required
rscales	tuple[float,...]|None	None
fay_rho	float|None	None
centering	Literal['full_sample','replicate_mean']	required

Field	Type	Default
generation	SurveyReplicateGeneration|None	None

columns
MinLen(min_length=2); MaxLen(max_length=500)
scale
Gt(gt=0.0)
fay_rho
Gt(gt=0.0); Lt(lt=1.0)

4.2.42 SurveyStage

tabelz.contracts.survey_design.SurveyStage

One sampling stage: its strata, its units and its FPC.

Field	Type	Default
strata	str|None	None
units	str|None	None
fpc	str|None	None
fpc_kind	Literal['count','fraction']	'count'

4.3 tabelz.data

4.3.1 CsvImportOptions

tabelz.data.models.CsvImportOptions

CSV import options used for previews and registration.

Field	Type	Default
delimiter	str	','
encoding	str	'utf-8'
decimal	str	'.'
header_row	int	1
first_data_row	int	2
data_types	dict[str,str]	factory:dict
missing_codes	tuple[str,...]	('',)
preview_row_limit	int	25
preview_column_lim it	int	20
preview_byte_limit	int	131072

4.3.2 CsvPreview

tabelz.data.models.CsvPreview

Bounded preview of a CSV source.

Field	Type	Default
source_path	str	required
raw_lines	tuple[str,...]	required
column_names	tuple[str,...]	required
parsed_rows	tuple[tuple[str,...],...]	required
byte_limit_reached	bool	required
warnings	tuple[str,...]	()

4.3.3 ImportResult

tabelz.data.models.ImportResult

Result of registering a source in the table catalog.

Field	Type	Default
table	ImportedTable	required
warnings	tuple[str,...]	required

4.3.4 ImportedTable

tabelz.data.models.ImportedTable

Registered source table backed by a lazy scan.

Field	Type	Default
table_ref	TableRef	required
source_path	str	required
import_options	ImportOptions	required
source_checksum	str	required
schema_fingerprint	str	required
logical_plan_fingerprint	str	required
scan_kind	str	required
row_count_estimate	int	required
column_names	tuple[str,...]	required
import_warnings	tuple[str,...]	required
lazy_frame	polars.lazyframe.frame.LazyFrame	required
scan_source_path	str|None	None
owns_scan_source	bool	False
owned_auxiliary_paths	tuple[str,...]	()
source_snapshot_at	str|None	None

Field	Type	Default
source_metadata	StatisticalImportRecord|OpenFormatImportRecord|None	None
server_details	tuple[tuple[str,str],...]	()

4.3.5 SqlImportOptions

tabelz.data.models.SqlImportOptions

Read-only relational query options used for preview and registration.

Field	Type	Default
database_kind	Literal['sqlite','duckdb','postgresql','sqlserver']	'sqlite'
query	str	''
parameters	dict[str,SqlParameter]	factory:dict
dataset_name	str	''
stable_key_columns	tuple[str,...]	()
data_types	dict[str,str]	factory:dict
preview_row_limit	int	25
preview_column_limit	int	20
query_timeout_seconds	float	30.0
maximum_import_rows	int	1000000
maximum_import_bytes	int	268435456
connection	SqlServerConnection|None	None

4.3.6 SqlPreview

tabelz.data.models.SqlPreview

Bounded preview of a read-only SQL query and its database catalog.

Field	Type	Default
source_path	str	required
table_names	tuple[str,...]	required
query	str	required
column_names	tuple[str,...]	required
parsed_rows	tuple[tuple[str,...],...]	required
byte_limit_reached	bool	required
warnings	tuple[str,...]	()

Field	Type	Default
suggested_data_types	dict[str,str]	factory:dict

4.3.7 SqlServerConnection

tabelz.data.models.SqlServerConnection

Where a database server is, without any credential (F158).

Field	Type	Default
host	str	required
port	int	required
database	str	required
user	str	required
tls	Literal['require','verify-full','disable']	'require'
odbc_driver	str	''

4.3.8 TableCatalog

tabelz.data.models.TableCatalog

In-memory table catalog keyed by public table identifier.

Field	Type	Default
tables	dict[str,ImportedTable]	factory:dict

4.3.9 OpenFormatImportRecord

tabelz.data.open_format_metadata.OpenFormatImportRecord

What an open-format import keeps with its table: the metadata mapping and provenance.

Field	Type	Default
mapping	OpenFormatMapping	required
provenance	tuple[tuple[str,str],...]	required

4.3.10 OpenFormatMapping

tabelz.data.open_format_metadata.OpenFormatMapping

F131 column metadata an open-format file declares (R factors, value labels, declared missing values and variable labels). “factor_columns“ names the R factors only; “special_missing“ lists the R columns with haven tagged missing values and their code columns, and “original“ records them with the file’s identity (only when there are any).

Field	Type	Default
source_format	str	required
columns	Mapping[str,ColumnMetadata]	factory:dict
factor_columns	tuple[str,...]	()
special_missing	tuple[SpecialMissingColumn,...]	()
original	ImportedFileMetadata|None	None

4.3.11 SourceSnapshotWrite

tabelz.data.source_snapshot_codec.SourceSnapshotWrite

A validated snapshot and its two package members before packaging.

Field	Type	Default
metadata	SourceSnapshotMetadata	required
members	dict[str,pathlib._local.Path]	required

4.3.12 DeclaredMissing

tabelz.data.statistical_metadata.DeclaredMissing

One SPSS missing-value declaration: a discrete value (low == high) or a range.

Field	Type	Default
low	LabelKey	required
high	LabelKey	required

4.3.13 MetadataMapping

tabelz.data.statistical_metadata.MetadataMapping

F131 column metadata for an imported file and the provenance of the mapping.

Field	Type	Default
source_format	str	required
columns	Mapping[str,ColumnMetadata]	required
special_missing	tuple[SpecialMissingColumn,...]	()
shared_label_sets	tuple[tuple[str,tuple[str,...]],...]	()
inferred_levels	tuple[str,...]	()
notes	tuple[str,...]	()
user_missing_converted	bool	False
original	ImportedFileMetadata|None	None

4.3.14 StatisticalFileMetadata

`tabelz.data.statistical_metadata.StatisticalFileMetadata`

Everything the metadata-only discovery step reads from one file.

Field	Type	Default
`format`	`str`	`required`
`format_version`	`str`	`required`
`encoding`	`str`	`required`
`encoding_declared`	`bool`	`required`
`file_label`	`str`	`required`
`table_name`	`str`	`required`
`row_count`	`int|None`	`required`
`variables`	`tuple[VariableCapture,...]`	`required`
`label_sets`	`tuple[tuple[str,tuple[tuple[LabelKey,str],...]],...]`	`()`
`catalog`	`str`	`''`
`library_version`	`str`	`''`
`notes`	`tuple[str,...]`	`()`

4.3.15 StatisticalImportRecord

`tabelz.data.statistical_metadata.StatisticalImportRecord`

What an SPSS, Stata or SAS import keeps with its table: metadata, mapping, provenance.

Field	Type	Default
`metadata`	`StatisticalFileMetadata`	`required`
`mapping`	`MetadataMapping`	`required`
`provenance`	`tuple[tuple[str,str],...]`	`required`
`integer_columns`	`tuple[str,...]`	`()`
`code_columns`	`tuple[tuple[str,str],...]`	`()`

4.3.16 VariableCapture

`tabelz.data.statistical_metadata.VariableCapture`

One variable as the file declares it.

Field	Type	Default
`name`	`str`	`required`
`label`	`str`	`required`
`storage_type`	`str`	`required`
`format`	`str`	`required`
`kind`	`str`	`required`

Field	Type	Default
storage_width	int|None	None
display_width	int|None	None
measure	str	''
label_set	str	''
declared_missing	tuple[DeclaredMissing,...]	()

4.4 tabelz.executor

4.4.1 RecipeReplayPayload

tabelz.executor.recipe_replay.RecipeReplayPayload

Complete unpublished recipe values carried back from the engine.

Field	Type	Default
tables	dict[tuple[str,int],polars.dataframe.frame.DataFrame]	required
results	dict[str,object]	required
exclusions	dict[tuple[str,int],frozenset[str]]	required
outcomes	dict[str,OperationOutcome]	required
operation_order	tuple[str,...]	required

4.5 tabelz.operations

4.5.1 AnalysisRecipeRequest

tabelz.operations.analysis_recipes.AnalysisRecipeRequest

An immutable analysis invocation captured before worker submission.

Field	Type	Default
operation_id	str	required
operation_type	str	required
schema_version	str	required
implementation_version	str	required
table_state	TableStateRef	required
result	ResultRef	required
input_results	tuple[ResultRef,...]	()
settings	JsonObject	required
population	PopulationRef|None	None
label	str|None	None
seed	int	required

operation_type
MinLen(min_length=1)
schema_version
MinLen(min_length=1)
implementation_version
MinLen(min_length=1)
seed
Ge(ge=0); Le(le=9223372036854775807)

4.5.2 GraphExecution

`tabelz.operations.execution.GraphExecution`

Lazy execution outputs indexed by their producing graph operation.

Field	Type	Default
`operations`	`Mapping[str,OperationExecution]`	`required`

4.5.3 OperationExecution

`tabelz.operations.execution.OperationExecution`

Lazy execution output and deterministic receipt data.

Field	Type	Default
`frame`	`polars.lazyframe.frame.LazyFrame`	`required`
`receipt`	`dict[str,Any]`	`required`

4.5.4 ColumnFormula

`tabelz.operations.formula_definition.ColumnFormula`

The persistent definition attached to one target column.

Field	Type	Default
`target`	`str`	`required`
`text`	`str`	`required`
`expression`	`dict[str,Any]`	`required`
`mode`	`Literal['numeric','text','date']`	`'numeric'`
`reference_date`	`str|None`	`None`
`formula_version`	`int`	`2`

4.5.5 CarriedColumn

`tabelz.operations.graph.CarriedColumn`

One output column copied unchanged from input columns (F131.2).

Field	Type	Default
column	str	required
sources	tuple[tuple[int,str],...]	required
dtype	str|None	None

4.5.6 MetadataCarry

tabelz.operations.graph.MetadataCarry

Marks a node recorded under column-metadata carry-over (F131.2).

Field	Type	Default
columns	tuple[CarriedColumn,...]|None	None

4.5.7 OperationGraph

tabelz.operations.graph.OperationGraph

Validated directed acyclic graph of operation nodes.

Field	Type	Default
nodes	tuple[OperationNode,...]	()

4.5.8 OperationNode

tabelz.operations.graph.OperationNode

A graph node containing one immutable operation spec.

Field	Type	Default
spec	OperationSpec	required
parent_operation_ids	tuple[str,...]	()
lifecycle_state	OperationLifecycleState	<OperationLifecycleState.READY:'ready'>
input_table_states	tuple[TableStateRef,...]	()
output_table_states	tuple[TableStateRef,...]	()
owner	str|None	None
metadata_carry	MetadataCarry|None	None

4.5.9 OperationDefinition

tabelz.operations.registry.OperationDefinition

Versioned description of a registered operation implementation.

Field	Type	Default
operation_type	str	required
schema_version	str	required
implementation_version	str	required
validate	Callable[[OperationSpec],None]	_noop_validate
outputs	OperationOutputContract	OperationOutputContractinstance
replay	Callable[[OperationNode,ReplayInputs],ReplayStep]|None	None

4.5.10 OperationRegistry

tabelz.operations.registry.OperationRegistry

Registry keyed by operation type and version tuple.

Field	Type	Default
_definitions	dict[tuple[str,str,str],OperationDefinition]	factory:dict

4.5.11 OperationVersionMigration

tabelz.operations.registry.OperationVersionMigration

An explicit, one-way upgrade for generated operation specifications.

Field	Type	Default
operation_type	str	required
schema_version	str	required
implementation_version	str	required
migrate	Callable[[OperationSpec],OperationSpec]	required

4.5.12 OperationVersionMigrationRegistry

tabelz.operations.registry.OperationVersionMigrationRegistry

Registry of supported historical operation versions for code generation.

Field	Type	Default
_migrations	dict[tuple[str,str,str],OperationVersionMigration]	factory:dict

4.6 tabelz.provenance

4.6.1 EngineProvenance

`tabelz.provenance.analysis.EngineProvenance`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`engine`	`EngineVersion`	`required`
`libraries`	`tuple[LibraryVersion, ...]`	`()`

4.6.2 ExclusionCount

`tabelz.provenance.analysis.ExclusionCount`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`reason_code`	`str`	`required`
`count`	`int`	`required`

count
Ge(ge=0)

4.6.3 InputFingerprint

`tabelz.provenance.analysis.InputFingerprint`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`role`	`str`	`required`
`fingerprint`	`str`	`required`

role
MinLen(min_length=1)

4.6.4 InvocationParameter

`tabelz.provenance.analysis.InvocationParameter`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`key`	`str`	`required`
`value`	`IntegerValue|FloatValue|NullValue|StringValue|BooleanValue`	`required`

key
MinLen(min_length=1)

4.6.5 LibraryVersion

`tabelz.provenance.analysis.LibraryVersion`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`name`	`str`	`required`
`version`	`str`	`required`

name
MinLen(min_length=1)
version
MinLen(min_length=1)

4.6.6 NoFilter

`tabelz.provenance.analysis.NoFilter`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`kind`	`Literal['none']`	`'none'`

4.6.7 OperationProvenance

`tabelz.provenance.analysis.OperationProvenance`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`operation_id`	`str`	`required`
`identity`	`OperationIdentity`	`required`

operation_id
MinLen(min_length=36); MaxLen(max_length=36)

4.6.8 PopulationProvenance

`tabelz.provenance.analysis.PopulationProvenance`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
`population_id`	`str`	`required`
`table_id`	`str`	`required`
`input_row_count`	`int`	`required`
`analyzed_row_count`	`int`	`required`

Field	Type	Default
filter	NoFilter|SelectionFilter|PredicateFilter	required

population_id
MinLen(min_length=36); MaxLen(max_length=36)

table_id
MinLen(min_length=36); MaxLen(max_length=36)

input_row_count
Ge(ge=0)

analyzed_row_count
Ge(ge=0)

4.6.9 PredicateExpression

tabelz.provenance.analysis.PredicateExpression

Strict V1 column/value predicate; paths, code, and callbacks are unrepresentable.

Field	Type	Default
column	str	required
operator	Literal['equals','not_equals','greater_than','greater_than_or_equal','less_than','less_than_or_equal']	required
value	IntegerValue|FloatValue|NullValue|StringValue|BooleanValue	required

column
MinLen(min_length=1)

4.6.10 PredicateFilter

tabelz.provenance.analysis.PredicateFilter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['predicate']	'predicate'
predicate_schema_version	Literal['1.0.0']	'1.0.0'
expression	PredicateExpression	required
expression_fingerprint	str	required

4.6.11 ProfileProvenance

tabelz.provenance.analysis.ProfileProvenance

Common strict, immutable model baseline for public contracts.

Field	Type	Default
profile	RegisteredRunProfile|CustomRunProfile	required
lineage	RegisteredLineage|ModifiedLineage|AdHocLineage	required

4.6.12 ProvenanceReceipt

tabelz.provenance.analysis.ProvenanceReceipt

Common strict, immutable model baseline for public contracts.

Field	Type	Default
schema_version	Literal['1.0.0']	'1.0.0'
sources	tuple[SourceInput,...]	required
population	PopulationProvenance	required
exclusions	tuple[ExclusionCount,...]	()
profile	ProfileProvenance	required
parameters	tuple[InvocationParameter,...]	()
operation	OperationProvenance	required
engine	EngineProvenance	required
input_fingerprints	tuple[InputFingerprint,...]	()

sources
MinLen(min_length=1)

4.6.13 SelectionFilter

tabelz.provenance.analysis.SelectionFilter

Common strict, immutable model baseline for public contracts.

Field	Type	Default
kind	Literal['selection']	'selection'
table_id	str	required
selected_count	int	required
membership_fingerprint	str	required

table_id
MinLen(min_length=36); MaxLen(max_length=36)

selected_count
Ge(ge=0)

4.6.14 SourceInput

`tabelz.provenance.analysis.SourceInput`

Common strict, immutable model baseline for public contracts.

Field	Type	Default
role	str	required
table_id	str	required
content_fingerprint	str	required
source_checksum	str|None	None
schema_fingerprint	str	required
logical_plan_fingerprint	str|None	None
row_count	int	required

role
MinLen(min_length=1)

table_id
MinLen(min_length=36); MaxLen(max_length=36)

row_count
Ge(ge=0)

4.7 tabelz.recipes

4.7.1 ResultHandle

`tabelz.recipes.api.ResultHandle`

A named result returned by a recorded analysis.

Field	Type	Default
ref	ResultRef	required
operation_id	str	required

4.7.2 TableHandle

`tabelz.recipes.api.TableHandle`

A stable revision of a table, independent of the active desktop tab.

Field	Type	Default
state	TableStateRef	required

4.8 tabelz.selection

4.8.1 EntityKey

`tabelz.selection.entities.EntityKey`

A persisted logical key over one or more columns in a table namespace.

Field	Type	Default
`key_id`	`str`	`required`
`table_namespace`	`str`	`required`
`columns`	`tuple[str, ...]`	`required`
`schema_version`	`str`	`'1.0.0'`

4.8.2 Selection

`tabelz.selection.models.Selection`

Compact row-ID membership; table rows are never copied.

Field	Type	Default
`table_namespace`	`str`	`required`
`row_ids`	`tuple[str, ...]`	`required`

Part II

Extension and implementation

CHAPTER 5

API: enterprise contract prototype

TBLZ-030 is an optional portability proof. It does not add a Hub or any networked enterprise feature.

5.1 Contract groups

All contract groups use schema version `1.0.0`, forbid unknown fields, and have a checked-in JSON Schema under `schemas/contracts/`.

- Repository: capability discovery plus bounded search, schema, preview, and content-addressed export.
- Recipe: a canonical operation graph, ordered repository source bindings, stable table identities, and an Ed25519-signed envelope.
- Worker: independently owned policy, a signed run request, data-free findings, per-operation receipt fingerprints, final rows, and a durable run receipt.

5.2 Local mock flow

The local proof keeps repository, recipe, and worker inputs separate:

1. `LocalMockRepository` creates a filesystem fixture and records immutable CSV metadata.
2. Desktop imports that object and produces its normal `OperationGraph` and foundation receipts.
3. `recipe_from_graph` serializes the same graph with repository bindings.
4. `sign_recipe` signs domain-separated canonical recipe bytes.
5. `python-mtabelz.enterprise.mock_worker` reads a request, a worker-owned policy, a worker-owned trust store, and the mock repository from separate files.
6. The worker validates trust, policy, graph roots, content fingerprints, and resource bounds before executing with `FoundationOperationRuntime`.

The integration fixture at `tests/integration/test_enterprise_contracts.py` launches that final step in a fresh Python process. It requires exact equality with the Desktop path for output rows, the final output fingerprint, and the final operation-receipt fingerprint.

5.3 Explicit non-goals

The prototype has no real authentication, remote transport, cloud repository, production scheduler, connector, notification system, marketplace, or MCP server. The local trust-store file is only a test boundary for signature verification; it is not a user or service identity system.

CHAPTER 6

Desktop data and display contracts

This chapter records implementation details that support the customer workflows. Statistical definitions belong beside their methods in the user manual; public Python and recipe entry points belong in its dedicated API appendix. These contracts do not imply that every underlying library option is available in the desktop interface.

6.1 Database imports

SQLite and DuckDB imports accept a single `SELECT` or `WITH` statement returning named, unique, non-reserved columns. Both connections are read-only. SQLite's authorizer denies mutation, transactions, attachment and pragma actions. DuckDB uses `read_only=True` and `enable_external_access=false`, rejecting attachment, extension installation/loading and file-reading table functions.

Parameters are scalar strings, numbers, booleans or null, with SQLite `:name` and DuckDB `$name` placeholders. Missing required parameters fail; unused provided values are ignored. DuckDB positional question-mark parameters are rejected. `SqlImportOptions.data_types` supports string, integer, float, Boolean, date, datetime, time and binary conversions with the accepted aliases.

Imports run in a spawned process, enforce row, byte and execution-time ceilings, and materialize an owned checksummed Parquet snapshot exposed through the normal lazy-frame path. SQLite uses a progress handler for timeouts; DuckDB uses a watchdog interruption. Stable keys are checked over the complete result. Without a declared key, row identities are snapshot-order identities and selection/exclusion metadata remain snapshot-scoped.

Source integrity includes the database file and any relevant write-ahead log: SQLite committed WAL content and DuckDB's `.wal` sibling. An active DuckDB writer can prevent import through its exclusive lock. Opening a project restores the saved snapshot without reconnecting; explicit refresh publishes atomically on success and advances the source revision. Failure retains the prior snapshot.

6.2 Derived table lineage

The registered `tabelz.join` operation retains both input references, the output reference, checksums, settings and stable row lineage. Stack and pivot retain source-row/measurement mappings and exclusions. These records support save, reopen, undo, redo and replay rather than inferring lineage from visible row numbers.

6.3 Scatter display allocation and facets

Grouped scatter displays share a budget of 5,000 marks. Groups are considered from smallest to largest; each receives the smaller of its valid population and an equal share of the remaining budget. Remainders are assigned in panel and colour order to groups with undrawn points.

Within a group, the smallest stable row-identity hashes with seed 0 are drawn. Full-population source, valid and displayed counts remain separate.

Facet layout considers columns with at least 220 pixels per panel, always permitting a single column. It prefers layouts with height at least 170 pixels within the visible area, breaking ties by the largest minimum of panel width and 1.5 times panel height. A preferred layout fills the visible height. Otherwise panel height is two-thirds of width, bounded below by 170 pixels and above by visible height, and scrolling replaces further shrinking. The legend wraps and shortens labels exceeding 160 pixels; the group summary retains full labels.

6.4 Correspondence-analysis execution

The desktop correspondence-analysis workflow has a 180-second execution deadline and a 64 MiB result limit. Cancellation and request-generation checks prevent obsolete results from replacing a newer request.

6.5 Saved row identities and score columns

Edited desktop source snapshots preserve exact row IDs in checksummed Parquet identity members alongside their CSV data. Optional `dataset_row_identities` UI-state metadata link these members to source checksums and population sizes. Missing, corrupt or mismatched identities fail restoration. Older projects without this metadata retain their established import behavior.

Retained PCA score operations store output names, source-table identity, means, scales, basis and fit-population row IDs. They do not refit. Creating columns rejects existing names. Replay against an already-created snapshot succeeds only if retained scores agree with the basis; conflicting values fail. Reports retain actual projected-input checksums, imported-source identity, fit-population fingerprints and source revision as well as bounded tables, plots, settings and warnings.

6.6 Structured Gaussian covariance estimation

Both modes are fitted by constrained maximum likelihood (ML) or restricted maximum likelihood (REML); REML is the default. The profiled deviance is evaluated by penalized least squares, as in lme4, with a stable factorization of centred data. The criterion for ML is $\log|I + \Lambda^{\mathsf{T}} Z^{\mathsf{T}} Z \Lambda| + N\{1 + \log(2\pi r^2/N)\}$. REML adds $\log|R_X|^2$ and uses $N - p$. Here r^2 is the penalized residual sum of squares, and $G = \sigma^2 \Lambda\Lambda^{\mathsf{T}}$.

The random intercept and slope covariance uses $\Lambda = \left(\begin{smallmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{smallmatrix}\right)$ with $l_{11}, l_{22} \geq 0$. This reaches every allowed covariance exactly: a positive-definite G; every rank-one G, with either sign of the covariance; a zero intercept variance with a positive slope variance; a zero slope variance; and $G = 0$. Each variance component is $\sigma^2\theta_k^2$ with $\theta_k \geq 0$.

Every boundary is fitted explicitly and compared with the interior. For variance components that means each subset of components at zero, including all of them. For the random slope it means the rank-one, zero-intercept and zero-covariance cases. Each case starts from fixed dispersed values. The fit with the highest likelihood over all attempts is kept. A zero variance in that fit is then checked by restarting with it small and positive, because at zero a factor parameter can look stationary even when a larger variance would fit better. Nothing is set to zero after fitting, so small positive variances are kept.

The fit with the lowest deviance is always the one reported. A tie within numerical tolerance is never resolved in favour of a smaller model, and no threshold depends on the response units.

A result is marked as a boundary estimate only when the selected fit lies exactly on a boundary and the first-order conditions for a constrained optimum hold there beyond their numerical error. At $G = 0$ the covariance score must be positive semidefinite in every feasible direction, not just the diagonal ones. On a rank-one covariance, the direction that raises the rank must not lower the deviance, and rotating within the face must be stationary. For a zero variance component, the one-sided derivative must be non-negative. These derivatives are extrapolated finite differences of deviance differences, computed so that they do not depend on the response units, and each comes with an error estimate. A feasible direction that lowers the deviance restarts the search from it; if that still fails, the fit is refused as not converged. An optimizer run that does not report convergence is refined and otherwise discarded, and the chosen fit must not improve along its numerical gradient.

When the sign of a derivative cannot be resolved, or when a threshold places the fit close to a boundary, the result is labelled *near boundary*. The threshold cases are a variance under 10^{-6} of the residual variance (after multiplying the slope variance by the mean squared slope value) and a nearly singular correlation. That label is a numerical diagnostic, not a test, and it never changes the estimates. A residual variance that cannot be resolved at working precision is reported as numerically unresolved, not as a singular likelihood.

When a random variance is zero, the intercept/slope correlation is undefined and is not reported as 0 or ± 1. Different covariance matrices that reach the same likelihood make the model unidentified, and the fit is refused. A likelihood that keeps increasing with the variance ratios (for example, toward a zero residual variance) or a search that reaches its limit is refused as unresolved. It is not reported as a boundary. This is a documented numerical maximization, recorded with the result. It is not a certificate of global optimality.

6.7 Generalized mixed-model numerical policy

Estimation. The fit maximizes the marginal likelihood (maximum likelihood, not REML or a Bayesian substitute). With $u = b/\tau$ each group's integral $L_g = \int \prod_i f(y_i \mid x_i^\top \beta + \tau u)\, \phi(u)\, du$ is evaluated by one-dimensional adaptive Gauss–Hermite quadrature centred at the posterior mode $\hat{u}_g$ with scale $\hat{\sigma}_g = (-h''(\hat{u}_g))^{-1/2}$:

$$\log L_g \approx \log(\sqrt{2}\,\hat{\sigma}_g) + \operatorname{logsumexp}_k(\log w_k + z_k^2 + h(\hat{u}_g + \sqrt{2}\,\hat{\sigma}_g z_k)),$$

computed with log-sum-exp. Binomial coefficients and $-\log y!$ are included, so log-likelihoods are complete. The fit starts at 15 nodes and refits at 31 and then 61; it stops when two successive node counts agree: the log-likelihood changes by at most 10^{-7} per observation and the fixed coefficients and τ change by at most 10^{-5} (scaled by $\max(1, |\cdot|)$). The higher-node fit is used. If 61 nodes still disagree with 31, the estimates are reported with a failed-accuracy status and no inference. Parameters are optimized as β on a centred-and-scaled design and $\log \tau$ with $\tau \in [10^{-4}, 10^3]$. The first node count is fitted from the GLM coefficients with three deterministic starting SDs, $\tau = 0.1$, 0.5 and 2; the best converged objective is kept and carried through the node schedule, and the results list each start. Starts that reach the same likelihood (within 10^{-6}) at materially different parameters make the optimization unresolved rather than accepted because one solution repeats across node counts. A fit is converged when the scaled gradient is at most 10^{-6} (a diagnostic) and every posterior-mode solve converged; before inference the optimum is also certified on the curvature scale, by a Newton decrement $g^\top H^{-1} g$ of at most 10^{-8}, which does not depend on additive likelihood constants. The zero-variance GLM fit reports its own convergence, certified the same way.

The zero-variance boundary. The ordinary GLM ($\tau = 0$) is fitted with the same likelihood and compared with the interior maximum; both log-likelihoods and the decision are recorded. The comparison uses an absolute tolerance equal to twice the verified quadrature accuracy of the objective (at least 10^{-8}), never a tolerance scaled by $|\log L|$. The interior fit is kept when it exceeds the boundary by more than this tolerance and is not at the lower search limit. A boundary estimate is reported only when the interior search is resolved (the optimizer converged and quadrature accuracy was verified), its likelihood does not exceed the boundary's by more than the verified tolerance, and the variance-scale score at zero,

$$\frac{\partial \log L}{\partial \tau^2}\Big|_{\tau=0} = \tfrac{1}{2}\sum_g \Big[\Big(\sum_i s_{gi}\Big)^2 + \sum_i h_{gi}\Big],$$

with s and h the first and second derivatives of the log-likelihood in the linear predictor at the GLM estimates, decides: when it is not positive the likelihood does not rise from $\tau = 0$, and the result reports $\tau = 0$ and the GLM fixed effects, so a boundary estimate is never concealed behind a small positive variance. The score is a local directional check: a non-positive value alone does not rule out a better positive-variance optimum, which is why the resolved interior comparison is also required. A failed or unverified interior search is not evidence that $\tau = 0$ is the maximum, and likelihoods are never described as agreeing within a verified tolerance when verification failed; the decision is then unresolved, with the provisional estimates identified. A positive-variance candidate at the lower search limit whose likelihood exceeds the boundary's by more than the tolerance is likewise unresolved, never a certified zero. Reaching the lower search limit (10^{-4}) is never by itself a reason to report $\tau = 0$. These tolerances are disclosed numerical policies, not accuracy theorems. When the score is positive but no interior maximum was resolved, when the SD reaches the upper search limit, or when the GLM fit did not converge, the decision is reported as numerically unresolved, the estimates are provisional and inference is withheld. Boundary Wald inference is not provided.

Inference. Fixed-effect standard errors come from the inverse observed Hessian of the joint marginal log-likelihood in $(\beta, \log \tau)$, so they include the variance parameter's uncertainty. The Hessian is formed by central differences of the analytic gradient at steps h and $h/2$ with Richardson extrapolation, at the retained node count; inference is withheld when the curvature changes by more than 10^{-3} (relative) between the two steps. Because both steps use the same approximate quadrature score, the extrapolated Hessian and the score are also recomputed with $2k + 1$ nodes (for k retained nodes); inference is withheld when the curvature changes by more than 10^{-3} or the Newton decrement there exceeds 10^{-8}. A Newton decrement counts only when its Hessian is finite and positive definite and the value is finite and non-negative; an indefinite Hessian can give a negative quadratic form, which is treated as a failed check. The analytic gradient holds each group's quadrature centring fixed, so it approximates the exact-integral score; the agreement of likelihoods and estimates across node counts does not by itself certify it, which is why the curvature-scale checks are applied. The $(\beta, \log \tau)$ parametrization is used at a regular interior optimum; a τ or τ^2 parametrization would not be intrinsically more correct. Tests use $z = \hat{\beta}/SE$ with two-sided normal p-values and intervals $\hat{\beta} \pm z_{1-\alpha/2} SE$. Odds ratios (binomial) or rate ratios (Poisson) are $e^{\hat{\beta}}$ with exponentiated interval endpoints; standard errors are never exponentiated. This is asymptotic normal Wald inference, not Kenward–Roger or another small-sample correction. Inference is withheld, with its cause, when the variance is at the boundary or unresolved, quadrature accuracy is not verified, the optimizer does not converge, the Newton decrement is too large, the curvature is step-sensitive, or the Hessian is not positive definite (no pseudoinverse is substituted). No reduced model is fitted in its place. The random-intercept SD is an estimate only: no test or interval is given for it.

6.8 GEE numerical conventions

Fisher scoring updates coefficients and exchangeable correlation for at most 100 iterations. The maximum scaled parameter change must be at most 10^{-8}, and the remaining scaled Fisher step and correlation update must be at most 10^{-6}. Failed convergence or a stalled finite iterate is provisional, with no inference or predictions.

The exchangeable Pearson scale is unresolved when RMS Pearson residual is at most 10^{-12}, or non-finite. Correlation is constrained to $[-1/(n_{\max}-1)+10^{-8}, 1-10^{-8}]$. These are operational numerical tolerances. The covariance definiteness check uses correlation scaling; an eigenvalue below -10^{-10} is material. Positive finite variances are not discarded by an absolute magnitude threshold. An exactly zero diagonal in an otherwise valid covariance requires zero cross-covariances and withholds that coefficient's inference.

The Mancl–DeRouen covariance and t_{G-p} convention are defined in the manual. Retained implementation cases `STAT-REG-GEE-001` and `STAT-REG-GEE-002` compare explicit base-R dense block calculations for binomial/exchangeable and Poisson/exposure/independence cases. Those implementation comparisons do not establish a comprehensive qualification campaign.

6.9 Repeated-covariance numerical conventions

An AR(1) schedule position is accepted within its floating-point computation error, at least 10^{-9} of a step. If that error exceeds 10^{-6} of a step, the schedule is ambiguous and refused. Correlation starts include 0, 0.5, -0.5 and 0.25; nonsingular candidates are optimized and the best converged objective retained. All-even supplied lags establish sign nonidentification; singular information at all tried starts is a numerical inability to establish identification.

For profiled GLS residuals r_i, the criteria are

$$-2\ell_{\mathrm{ML}} = \sum_i \log|V_i| + \sum_i r_i^{\mathsf{T}} V_i^{-1} r_i + N\log(2\pi), \qquad -2\ell_{\mathrm{REML}} = -2\ell_{\mathrm{ML}} + \log|C^{-1}| - p\log(2\pi).$$

Covariance parameters use an analytic gradient and bounded quasi-Newton optimization. A random-intercept zero boundary is explicitly compared.

For $P = V^{-1} - V^{-1}XCX^{\mathsf{T}}V^{-1}$ and $V_a = \partial V/\partial\theta_a$, information is

$$I_{ab} = \tfrac{1}{2}\operatorname{tr}(PV_aPV_b) \quad \text{(REML)}, \qquad I_{ab} = \tfrac{1}{2}\operatorname{tr}(V^{-1}V_aV^{-1}V_b) \quad \text{(ML)}.$$

The declared approximate covariance of natural covariance parameters is $A = I^{-1}$, with approximate standard errors from its diagonal, withheld at boundaries or singular information. For a fixed contrast,

$$\frac{\partial(c^{\mathsf{T}}Cc)}{\partial\theta_a} = c^{\mathsf{T}}CX^{\mathsf{T}}V^{-1}V_aV^{-1}XCc.$$

Together these define the manual's Satterthwaite calculation. Retained `STAT-REG-REPCOV-001` and `-002` check the OLS and balanced random-intercept degrees-of-freedom limits as implementation acceptance; they are not a comprehensive external-reference qualification.

6.10 Generalized mixed-model evidence scope

Implementation comparisons cover direct integration of fixed-parameter likelihoods, finite-difference gradients, zero-variance GLM limits, single-trial/grouped-binomial equivalence and exposure-unit shifts. `STAT-REG-GLMM-001` and `-002` retain closed-form boundary cases. These checks are separate from any later lme4 qualification.

6.11 Random-intercept profile details

For variance ratio $\lambda = \sigma_b^2/\sigma_e^2$, fixed effects and residual variance have generalized-least-squares profile solutions. Balanced intercept-only REML uses its exact constrained solution; ML does not. Other fits search $\log\lambda \in [-23, 23]$ on 2,001 points, refine local peaks, and refine any peak between zero and the first grid point on the linear scale. A second 4,001-point search must choose the same peak and agree within 10^{-9} log-likelihood per residual dimension ($N - p$ REML, N ML). An increasing upper edge or failed agreement is unresolved. This finite search is not a global-optimality certificate.

Centred/reference-level data and a within-group QR separate small within variation from between-group spread. Zero within residual variation gives unbounded REML when $d_w = N - G - \mathrm{rank}(X_{\mathrm{within}}) > 0$, and unbounded ML when $N > G$. REML with $d_w = 0$ retains a finite profile. Positive within residuals are assessed against roundoff of within deviations, not between-group spread. A winning zero boundary is recomputed exactly, with REML residual variance $\mathrm{RSS}/(N - p)$. Old saved results retain their `lbfgs` optimizer identity.

6.12 Factor-analysis boundary verification details

The orthogonal maximum-likelihood fit uses the locked statsmodels `Factor(method='ml')` implementation and a pinned start seed. A reported uniqueness at or below zero is a Heywood boundary. A positive uniqueness, however small, is not by itself evidence of that boundary.

If the optimizer fails, returns a nonpositive uniqueness, or reaches 10^{-10} or less, and at least one uniqueness is below 0.005, a zero-uniqueness candidate is solved using $\log\det\Sigma + \mathrm{tr}(R\Sigma^{-1})$. Certification requires vanishing free gradients, no improvement by increasing a zero uniqueness, and an objective no worse than the original fit. Only a verified candidate changes the status to Heywood. Otherwise the original nonconvergence remains, with an unresolved-boundary warning. An admissible converged solution below 0.005 receives a near-boundary advisory. Exact zero model degrees of freedom do not guarantee an admissible interior correlation solution.

6.13 Survival execution and display

`tabelz.executor.survival` runs a checksummed Parquet projection of identities and role columns after row-limit preflight. Desktop generation tokens suppress cancelled and superseded results. Display limits are 4,000 steps and 1,000 censor marks across groups, and 200 life-table rows; reductions are disclosed. There is no table mutation or registered transformation. Saved reports retain method/module versions, input checksum, source identity/revision, population fingerprint, settings and warnings. The existing censored-Weibull diagnostic remains a separate workflow.

6.14 Regression numerical dependencies

MANOVA supplies explicit term contrasts and an identity response transform to statsmodels. At an exactly null term, statsmodels 0.15 can encounter an empty positive-eigenvalue array while preparing the unused Roy statistic. HypoNum evaluates the same Pillai/Wilks approximation formulas at their null limits, retaining valid positive degrees of freedom and $p = 1$. This is neither covariance jitter nor a replacement test.

OLS, ANOVA and classical coefficient/mean-response inference use statsmodels. HypoNum computes HC3 and CR1 covariance, related t inference and joint F Wald tests with NumPy and SciPy. Scaling, KNN imputation, splitting, cross-validation, Ridge/Lasso and fit metrics use

scikit-learn; Q–Q and compiled neighbour averaging use SciPy. Functional comparisons against these libraries and direct matrices are distinct from independent qualification evidence.

VIF uses a rank-revealing QR projection of each centred encoded predictor on an intercept and the other columns. A residual below 10^{-12} of the centred norm is unresolved (corresponding to VIF above 10^{24}). Before centring, a column is scaled by its largest absolute value; constant detection compares variation with centring roundoff rather than an absolute-value threshold. This preserves units/origin invariance at the available numerical precision.

Ordinal and multinomial separation checks use joint-likelihood linear programs. A candidate direction is verified against the encoded design, with exact power-of-two scaling only when no input underflows. Uncertain slack signs are resolved by exact rational projection onto their zero constraints, then exact signing of every slack. Certification requires no negative slack and at least one positive slack, including threshold-order constraints for the ordinal case. Failed verification or unsafe scaling produces an unresolved check, not a separation claim.

For NB2, the Poisson-boundary score at the Poisson estimate is $\frac{1}{2}\sum_i[(y_i - \hat{\mu}_i)^2 - y_i]$. When fitted dispersion is tiny and the Poisson likelihood attains the NB2 likelihood, a score below minus its numerical error certifies a strict local maximum at $\alpha = 0$ over $\alpha \geq 0$. Error includes the summation rounding bound and the score change implied by the remaining Poisson Newton step. A score within that error leaves the boundary unresolved. Either outcome withholds ordinary NB2 Wald covariance. The joint observed-information convention differs from `MASS::glm.nb`'s expected IRLS information conditional on fitted $\theta = 1/\alpha$.

6.15 Nonlinear regression execution and numerical policy

Bounded trust-region reflective least squares uses SciPy's linear loss, Jacobian scaling and cost/step/gradient tolerances 10^{-14}. Forward-mode automatic differentiation evaluates the validated expression's Jacobian. Non-finite trial points are rejected as failed steps; starts must themselves be finite. Gradual underflow is allowed.

After convergence, SVD Gauss–Newton steps refine the exact gradient. A step must reduce gradient norm while keeping SSE within 10^{-12} relative to the trust-region phase's endpoint. Rank uses singular values relative to the largest with threshold $\max(\epsilon \max(n, p), 10^{-10})$. Full-rank condition above 10^{10}, or an active bound, withholds covariance. The SVD covariance calculation has rounding sensitivity proportional to the Jacobian condition at this step, separate from sensitivity of the underlying inferential problem. For mean-response gradients, a bounds-aware central difference is available when an analytic derivative is undefined.

Value and derivative calls are separately counted in total model evaluations; the additional model-call ceiling is budget times $2p + 2$. Restore artifacts are limited to 64 KiB, responses to 64 MiB, workers to 180 seconds and report blocks to 4 MiB. The service owns request generations, source snapshots, cancellation and stale suppression; project transitions clear session calculations. Preparation cancels current work while retaining a usable fit. Worker controls are versioned JSON with checksummed projected Parquet, without pickle or executable expressions. Saved reports retain strict configuration, usable covariance and fit-data/exclusion fingerprints alongside source identity and revision.

6.16 Deming execution and numerical policy

The line formula uses an algebraically equivalent form to avoid subtractive cancellation. Leave-one-out moments are derived from full centred sums in linear work; a deletion that consumes most of a moment is directly refitted under a work limit. No invalid deletion is omitted from a

jackknife interval. Paired percentile resampling uses NumPy `default_rng(seed)` and discloses valid-replicate conditioning. The worker has a 180-second and 64 MiB response limit. Saved semantic tables and the plot retain effective settings, source and exclusion fingerprints, method identity and implementation version, with no executable estimator object.

6.17 Multiple-imputation numerical policy

Each imputation k has its own NumPy `SeedSequence(seed, spawn_key=(k,))`. Initial fills sample observed values. Incomplete variables are visited by ascending missing count, then name, independent of physical column order.

Predictive mean matching fits $X_{\text{obs}} = QR$, requiring full rank and $n_{\text{obs}} > p$, then draws

$$s_*^2 = \text{RSS}/\chi^2_{n_{\text{obs}}-p}, \qquad \beta_* = \hat{\beta} + s_* R^{-1} z, \quad z \sim N(0, I).$$

The zero-RSS boundary uses $\beta_* = \hat{\beta}$ and discloses the loss of proper imputation uncertainty for a deterministic relation. Each missing-row draw is compared with observed-row predictions at $\hat{\beta}$. One of the five nearest donors is chosen uniformly. All ties, including at the fifth distance, are randomized over every tied observed row. Passive derived terms are unsupported.

Categorical imputation uses a reference-category logit with numeric predictors standardized on the target's observed rows, independent $N(0, 2.5^2)$ slope priors and $N(0, 5^2)$ intercept priors. Newton line search allows 100 iterations and step tolerance 10^{-9}. Acceptance additionally requires decrement $g^\mathsf{T} H^{-1} g \le 10^{-10}$, positive-definite Hessian and condition at most 10^{12}. For $H = LL^\mathsf{T}$, the Laplace draw is $\theta_* = \hat{\theta} + L^{-\mathsf{T}} z$. At least two observed categories are required for an incomplete categorical variable. Ordinal imputation currently uses the nominal model.

Completed-data logistic analysis is unpenalized IRLS, with 100 iterations, verified absence of separation, positive-definite information and condition at most 10^{12}. Separation verification is unavailable above 500,000 rows. Inconclusive verification or any failed completed-data analysis stops the study; successful completions are never selectively pooled. The source ceiling is one million rows. Numeric observed values must be exactly representable in Float64, including the integer-domain check.

6.18 Documentation baseline and retained performance context

The September 2026 documentation baseline records NumPy 2.5.2, pandas 3.0.5, Polars 1.44.1, scikit-learn 1.9.0, SciPy 1.18.1 and statsmodels 0.15.0 in the dependency lock. These identify implementation context; an upstream package's full interface is not thereby exposed in the desktop application. The correlation, fit-Y-by-X, logistic, specification, SPC, gage and attribute-agreement implementations and their desktop callers determine the supported options.

An earlier manual reported approximately 15 seconds and 2 GB for a 60-column principal-component analysis of 300000 rows (18 million cells) on a desktop computer. That statement did not identify the machine, measurement protocol or retained run artifact. It is preserved here as an unqualified historical performance observation, not as verified benchmark evidence or a customer performance guarantee. A release claim requires an identified source baseline, workload, machine and retained measurement record.

CHAPTER 7

Time-series estimation and reconstruction

The following numerical conventions support the customer descriptions of time-series methods. They are retained here for engineering review, reproduction and compatibility assessment.

ARIMA and seasonal ARIMA models, and regression with ARIMA errors, are fitted by exact Gaussian maximum likelihood in state-space form. The $d + sD$ integrated states start from an exact diffuse initialization and the ARMA states from their stationary distribution; no large-variance approximation ("κ") enters the likelihood. The exact diffuse filter adds, for each observation that resolves a diffuse direction, the parameter-free constant $-\frac{1}{2}\log(2\pi F_{\infty,t})$, where $F_{\infty,t}$ is that observation's diffuse forecast variance. The reported log likelihood removes exactly those constants and keeps every other contribution, including any parameter-dependent term inside the diffuse phase. For a complete series this is the exact likelihood of the differenced series $(1-B)^d(1-B^s)^D y_t$, the same quantity R's `arima` reports as its likelihood, but with the integrated states treated exactly. AIC is $-2\ell + 2k$ and BIC is $-2\ell + k\log n_{\text{used}}$, where k counts the ARMA, trend and innovation-variance parameters (not the diffuse states), and n_{used} is the number of observed values less the $d + sD$ observations that resolve the diffuse states. This initialization was introduced as an explicit method change (time-series module version 1.1.0, forecast implementation 1.2.0); earlier releases used an approximate diffuse prior with variance 10^6 and the first $d + sD$ observations burned, so estimates of integrated models saved before the change can differ slightly, and such saved results keep their original values and method record.

In rolling validation, MASE scaling comes from the training data of each origin, and estimated transformations and automatic selection are refitted inside every training window. Back-transformation of a cumulative transformation uses the joint forecast-error covariance and seeded forecast paths, so multi-step intervals reflect covariance across horizons. A mean forecast averages the back-transformed paths. A median forecast is the marginal median of the reconstructed level at each horizon, conditional on the fitted model; the medians of successive horizons need not form one realisable path. Where the reconstructed level is a monotone function of a single linear combination of the Gaussian forecast — ordinary differencing, detrending, seasonal adjustment, a log, log returns, or the first period of a simple return — the back-transformed centre is exactly that median. Elsewhere, such as a product of simple returns over several periods or differencing followed by a log (a sum of lognormal increments), the median is estimated from the back-transformed paths. Simulated levels and interval endpoints come from eight independently scrambled Sobol sequences with a power-of-two number of paths each. The spread between the eight estimates gives each horizon a numerical standard error, which is separate from the forecast's own uncertainty. For the study's future forecast the path count doubles, from 4,096 up to 32,768 per scramble (fewer on long horizons, to bound memory), until that error is at most 0.1 percent of the interval width. Horizons that miss this within the budget are reported with a warning that their precision is not established. Every simulated value keeps only the digits its own error supports, and the result records the method, scrambles, seeds, path count and error per horizon. The fixed seeds make a study replay exactly; they do not make it

more accurate. Rolling-origin forecasts use the initial path count. Annualized simple returns and annualized growth rates return to levels through a fractional power that exists only above −100 percent; a Gaussian forecast with any uncertainty assigns positive probability below that, so such a forecast is refused rather than approximated. Forecast log returns instead, whose annualization is linear. Computing annualized returns from observed data is unaffected, and an unannualized simple-return forecast can reach negative levels. Reports separate the original, effective transformed, and final-fit training counts. Final evaluation materialises the vintage available at its cutoff and forecasts through any publication gap before scoring its target horizon.

With an information-availability specification, each origin sees only the releases and revisions published by that origin, while evaluation values use the latest supplied vintage. The data fingerprint covers the supplied vintage history; the specification fingerprint covers the calendar, transformation, validation, estimation, and information policy. Dynamic-regression forecasts are labelled conditional on the supplied predictor path; archived scenarios can be restricted to those published by each origin, and choosing realised predictor paths instead labels all validation output as conditional hindsight.

GARCH fits retain their internal estimation scale while reporting dimensional parameters and volatility in the supplied return units. GARCH rolling scores use residual-variance forecasts and squared innovations centred on the fitted conditional mean, and the training-window benchmark follows the same convention. The fit uses the locked `arch` implementation. Returns are first multiplied by a power of ten chosen so that the variance of the preliminary residuals (returns less their sample mean for a constant mean, the returns themselves for a zero mean) lies in $[0.1, 10^4)$; that estimation scale is reported, and parameters are converted back to return units. The variance recursion is $\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i e_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$: the order p counts the lagged squared innovations (the α terms) and q the lagged conditional variances (the β terms), the convention of the locked `arch` release, and the parameters are listed as $\alpha_1, \ldots, \alpha_p$ and then $\beta_1, \ldots, \beta_q$. Presample squared innovations and variances are the fixed backcast $b = \sum_{i=0}^{m-1} 0.94^i e_i^2 / \sum_{i=0}^{m-1} 0.94^i$ over the first $m = \min(75, n)$ preliminary residuals, so a GARCH(1,1) starts at $\sigma_1^2 = \omega + (\alpha + \beta)b$, not at b. Every observation enters the Gaussian or standardized Student-t likelihood, which is maximized to an optimizer tolerance of 10^{-12}; parameter standard errors are the robust sandwich covariance. Each rolling training window builds its own backcast from its own observations only. The VAR lag control counts lags in the level VAR representation, so a rank-zero or partial-rank system fits one fewer lag of differences, including the valid zero-difference-lag boundary at one level lag. Impulse responses disclose their dependence on variable ordering, and Granger results describe incremental predictability only.

Named exchange calendars come from `pandas-market-calendars`. A calendar records its name and time zone, supplies regular and special sessions, separates expected closures from missing sessions, and generates future session dates. Every advanced result records its complete estimation specification, a schema version, data and specification fingerprints, and the Python, ARCH, NumPy, Polars, SciPy, and statsmodels versions that produced it. Saved-study comparison marks a differing target, unit, validation horizon, rolling sample, or final sample as incompatible rather than comparing them.

CHAPTER 8

Reliability numerical and desktop contracts

These details were relocated from the desktop manual. Public entry points remain in its dedicated Python appendix; the customer chapters explain observation models, settings, interpretation and examples.

8.1 Censored log-location-scale fitting

The right-censored Weibull, lognormal and loglogistic fits and mixed-censoring families use $\log T = \mu + \sigma W$. Smallest-extreme-value W gives Weibull and, with $\sigma = 1$, exponential; normal and logistic W give the other families. Positivity of dispersion is the only parameter restriction. The right-censored exponential scale has its separate analytic total-time-on-test calculation.

Likelihood is evaluated in centred log time, with location measured in dispersion units. Interval contributions use exact interval width and stable log-CDF/log-survival differences. For wide positive intervals, $\log(U/L)$ is evaluated as $\log U - \log L$ once $U > 2L$; probability increments remain in logarithmic form.

Acceptance is evaluated at the reported point. A heuristic inspired by Moré and Wild estimates evaluation noise from sixth differences on one fixed stencil, with a floor at value rounding. It is neither their complete procedure nor a bound on evaluation error. Trial moves of 0.1, 1 and 3 in fitted coordinates must not improve likelihood by more than ten times the estimated error. Observed information must be positive definite. A Richardson central-difference score, with its own error estimate, must satisfy $g^{\mathsf{T}} C g \leq 10^{-5}$, where C is inverse information. This Newton-decrement condition corresponds to the default scaled-gradient tolerance referenced to Stata's `ml`. It is approximate local convergence, not proof of a global maximum. Failure of the score condition alone means unresolved convergence, not proof that a finite maximum does not exist.

Advisory thresholds are shape greater than 50 for Weibull/loglogistic and lognormal dispersion below 0.02. They do not cap a resolved fit. Right-censored implementation 1.1.0 superseded historical bounds of shape 0.05–100, scale 10^{-12}–10^{15} and lognormal dispersion 0.01–10. Saved earlier results retain their original values and method record.

Accelerated-life fitting centres and scales transformed stress and maps estimates, likelihood and covariance back to the declared representation. Acceptance uses the final recentred point and the same resolved-fit checks. Parameter covariance is ordered as $(\beta_0, \beta_1, \log \text{dispersion})$.

8.2 Display and reporting limits

The right-censored desktop worker requests at most 400 diagnostic points. The optional public `max_diagnostic_points` must be at least two. Failed families remain identified; best-by-AIC considers resolved candidates. Mixed-censoring results have a fit table and censoring summary, without a nonparametric curve.

Accelerated-life curves use 101 geometrically spaced times, starting at 0.1 times the minimum observed lifetime and extending to a bounded upper display range at least twice the maximum observed lifetime. Fit, stress-diagnostic and survival-interval report tables display at most 200 rows and identify truncation. The payload retains each complete bounded 101-point family curve. A plot omits its band if any curve point has unsupported bounds, preserving the fitted curve and available table values.

Each report block has a 4 MiB payload ceiling, with at most 128 blocks per report and 32 reports per project. Oversized results fail visibly. Reliability configuration/result dataclasses have no generated public Pydantic schema snapshots.

8.3 Desktop execution and retained results

The three table-backed life workflows project selected columns and stable row identity only, with ceilings of 1,000,000 rows and 20,000,000 selected cells. Their disposable workers have a 180-second deadline and a 16 MiB JSON message limit. Temporary artifacts are cleaned after completion or failure. Cancellation, supersession, project/source changes and exclusion changes prevent stale publication. Closing a setup window alone does not cancel a running life-family worker.

Reportability requires source identity, revision and exclusions to match the ready calculation. Blocks retain selected roles, effective settings, source fingerprints, exclusions, warnings, method identity and implementation version with the numerical snapshot. Save/reopen displays the retained result without refitting. A rerun is an explicit new fit. These workflows do not mutate columns and have no table undo step.

8.4 Reliability planning duration terminology

The exponential failure-terminated planner returns `expected_duration` as required total time on test divided by the declared sample size. No replacement/risk-set model establishes this as an unconditional expected calendar stopping time under truncation at the next disallowed failure. The customer manual therefore describes it as a planning conversion. This wording limitation is retained here so the result-field name is not mistaken for an established expectation. Precision simulation scans every candidate sample size upward; it is not a monotone bisection. Its first passing size is conditional on the recorded seeded Monte Carlo estimate.

CHAPTER 9

Local analysis plug-in authoring

This is the developer contract for trusted local analysis plug-ins. Installation, user controls and trust decisions are described in the customer manual. The examples below retain the existing supported contract.

9.1 Authoring a plug-in

Local analysis plugins add an analysis setup form and bounded result views without changing HypoNum itself. A plugin is one folder containing a JSON manifest and a Python entry point. Plugin Python may use ordinary Python libraries; importing `tabelz` is not part of the plugin contract.

The contract is schema-driven: Python declares standard controls and result components and HypoNum renders them consistently. Plugins cannot inject arbitrary QML, Qt widgets, or HTML.

9.2 Sample and installation

The bundled **Sample: Column summary** plugin appears under **Plug-ins**. Import a table, open the sample, choose a numeric column, and select **Run analysis**.

For a local plugin, create one direct-child folder in the platform plugin root.

- Windows:
  `%LOCALAPPDATA%\HypoNum\plugins`
  `\my_plugin`.
- macOS: `~/Library/ApplicationSupport/Tabelz/plugins/my_plugin`.
- Linux:
  `$XDG_DATA_HOME/ tabelz/plugins`
  `/my_plugin`, falling back to `~/.local/share/ tabelz/plugins`
  `/my_plugin`.

During source development, `BEYONDSTAT_PLUGIN_PATH` may contain additional plugin roots separated by the platform path separator. Choose **Plug-ins** > **Reload plug-ins** after changing a manifest. Opening a plugin starts a fresh worker, so Python source changes are picked up on the next open or run.

9.3 Folder and manifest

The required folder shape is:

```
my_plugin/
  tabelz-plugin.json
  plugin.py
```

The manifest has this shape:

```
{
  "schema_version": "1.0.0",
  "plugin_id": "com.example.column_range",
  "name": "Column range",
  "version": "1.0.0",
  "description": "Calculate the finite range of one numeric column.",
  "entry_point": "plugin.py",
  "author": "Example author"
}
```

The plugin ID is stable and namespaced. The version uses three numeric components. The entry point must be a non-symlink `.py` file directly inside the plugin folder. Discovery reads only this manifest and does not import `plugin.py`.

9.4 Declarative setup form

The entry point defines `describe()` and returns a plain dictionary:

```
def describe():
    return {
        "schema_version": "1.0.0",
        "title": "Column range",
        "description": "Calculate the range of finite values.",
        "fields": [
            {
                "field_id": "column",
                "label": "Numeric column",
                "kind": "column",
                "numeric": True,
                "help": "Missing and non-finite values are excluded.",
            },
            {
                "field_id": "show_count",
                "label": "Show valid row count",
                "kind": "boolean",
                "default": True,
            },
        ],
    }
```

Supported field kinds are `column`, `number`, `integer`, `boolean`, `choice`, and `text`. Numeric fields may declare `minimum` and `maximum`; choice fields declare `choices`; and column fields may set `numeric` to `true`. HypoNum revalidates fields before execution.

9.5 Run contract and result components

The entry point also defines `run(tables,parameters)`. The `tables["active"]` value is a Polars `LazyFrame` for the active table. A second key uses the table's display name when possible. Plugins should collect only the columns and aggregates required by the analysis. This sample calculates a finite numeric range and optionally emits its valid-row count:

```
import math
```

```
import polars as pl


def run(tables, parameters):
    column = parameters["column"]
    summary = (
        tables["active"]
        .select(
            pl.col(column).filter(pl.col(column).is_finite()).min().alias("minimum"),
            pl.col(column).filter(pl.col(column).is_finite()).max().alias("maximum"),
            pl.col(column).filter(pl.col(column).is_finite()).count().alias("count"),
        )
        .collect()
        .row(0, named=True)
    )
    if summary["count"] == 0:
        raise ValueError(f"{column!r} has no finite numeric values.")
    data_range = float(summary["maximum"] - summary["minimum"])
    if not math.isfinite(data_range):
        raise ValueError("The range is not finite.")
    components = [
        {
            "component_type": "scalar",
            "key": "range",
            "label": "Range",
            "value": data_range,
            "units": None,
        }
    ]
    if parameters["show_count"]:
        components.append(
            {
                "component_type": "scalar",
                "key": "count",
                "label": "Valid rows",
                "value": int(summary["count"]),
                "units": None,
            }
        )
    return {
        "schema_version": "1.0.0",
        "title": f"Column range: {column}",
        "summary": f"Calculated the finite range of {column!r}.",
        "components": components,
        "warnings": [],
        "method_note": "Range equals maximum minus minimum.",
    }
```

Supported result components are:

- `scalar`: a label, JSON scalar value, and optional units.
- `table`: a title, unique columns, and rectangular JSON-scalar rows.
- `histogram`: labels, description, source and valid counts, and up to 64 bins.
- `scatter`: labels, description, source and valid counts, and up to 5,000 points.

A result has at most 32 components. A table has at most 50 columns, 1,000 rows, and 20,000 cells. A worker response is limited to 5 MiB. Strings, numbers, booleans, and null are safe table-cell values. Non-finite numbers, unknown fields, malformed counts, and duplicate component keys are rejected. Standard output and standard error are displayed in a bounded diagnostics section.

9.6 Execution and trust model

HypoNum materializes only the active table to a temporary Parquet file, hashes it, and opens it in a fresh spawned Python process. Plugin source is also hashed before import. Results are validated in both worker and Desktop processes; late results cannot publish into a different dataset tab. A run can be cancelled or time out after 120 seconds.

The result footer shows plugin identity, version, and abbreviated code checksum. The process boundary is not a sandbox: a plugin has the same filesystem, process, and network permissions as the signed-in user. Review and trust source before running it. HypoNum does not install plugin dependencies; a plugin can use packages already present in the HypoNum runtime. Projects do not embed plugins or run them on open.

CHAPTER 10

Planning and retained-model numerical contracts

10.1 Prospective calculations and transport

Precision and extended-power calculations run from entered parameters and retain a canonical request SHA-256 fingerprint, effective configuration, method identity, implementation version and typed result. Their source declaration is parameter-only, with no fictitious table, revision, row membership or data fingerprint. Incoherent family fields fail explicitly. In normal-variance power planning the prospective field is `variance_ratio`; the inference field `null_variance` is refused.

TOST power integrates the joint acceptance event over the bounded chi-square probability range by adaptive quadrature. Absolute and relative tolerances are 10^{-10} and the reported integration error must be at most 10^{-8}. Mean, proportion, Z, ANOVA and variance sample-size solves use bounded monotone brackets and bisection with integer rechecks; Poisson and TOST use ascending search with a 20000-evaluation budget. The supported effect solves are mean, proportion and Z only.

Parameter-only reports preserve the complete typed setting/result pair and reject inconsistent or fictitious source fields. The additive extended-power report behavior keeps the existing report schema version; older application versions can reject these snapshots on downgrade. Curve tables retain 200 display rows while the bounded numerical result can retain 201 points. Calculation deadlines are 180 seconds; cancellation and request generations suppress superseded publication.

10.2 Stable scoring covariance and influence

OLS/WLS coefficients use QR decomposition of the weighted or unweighted design. The frozen fit stores $F = sR^{-1}$ with $\mathrm{Cov}(\hat{\beta}) = FF^{\mathsf{T}}$. For logistic and Poisson, $F = R^{-1}$ from the QR decomposition of $W^{1/2}X$. Prediction variance is evaluated as $\|x^{\mathsf{T}}F\|^2$, avoiding cancellation in an expanded quadratic form at large predictor offsets. Earlier artifacts lacking this factor retain covariance-based scoring compatibility.

OLS influence diagnostics use response/design centring with an intercept. Translating response leaves the diagnostics invariant. Translating predictors leaves residuals, leverage, standardized/deleted residuals, Cook's distance and DFFITS invariant. DFBETAS follow the coefficients of the specified parameterization; changing the intercept definition changes its DFBETAS. A deletion is described as determining a coefficient alone only when it reduces design rank. Quantities requiring unresolved $1 - h_i$, s^2 or deleted variance are individually withheld at the centred-computation rounding scale, without adding a small surrogate.

Scoring input has 1000000-row and 20000000-selected-cell limits; encoded models have at most 32 predictors, 32 categorical levels per predictor and 256 features. Bounded fit output is 64 MiB, so an influence payload can exceed it before the row ceiling. The calculation deadline is

180 seconds. Reports have the existing 4 MiB card limit and retain 200 display rows with the complete frozen artifact; the live influence plot is not the exported report score table.

10.3 Stored profiler execution

Model bindings, full fit artifacts, fitting membership, settings and result fingerprints are validated together. The scenario report must not claim the active table as its fitted population. Messages have a 64 MiB bound and 180-second worker deadline. Cancellation, closure, supersession and project/dataset reset suppress late generations without altering fits or data. All categorical combinations are enumerated; numeric uniform and local budgets are apportioned across them. The held setting and all-lower/all-upper corners add evaluations beyond the nominal uniform candidate budget. Profile points are not counted as optimization evaluations. Ties preserve the first best setting.

New scoring fits retain numeric ranges from complete, non-excluded fitting rows. Older frozen fits remain valid without those optional ranges. Explicitly declared fitted ranges can be provided to the typed profiler configuration; absent evidence cannot certify extrapolation.

10.4 Short-run monitoring transport

Short-run Z–MR desktop workers have a 180-second deadline and a 64 MiB response limit. This execution information belongs to engineering rather than the statistical interpretation of standardized observations.

CHAPTER 11

Measurement-system contracts and reference conventions

The user manual describes study design, statistical definitions and the desktop workflow. This chapter retains serialization details and the historical comparison decisions that support the shipped profiles. It is engineering evidence, not a customer qualification claim.

11.1 Standard identity and transport

The built-in identifiers are `aiag-msa-4` and `tabelz-extended`, initially version 1. A setting fingerprint is the SHA-256 of canonical JSON containing settings and permitted per-study choices. The user library is `\%LOCALAPPDATA\%/Tabelz/standards`, with one `<identifier>-v<version>.json` file per version.

A UTF-8 file uses format `tabelz-gage-standard`, schema version 1, and the `gage-standard-profile` public schema. It retains identifier, name, version, documentation reference, lineage, base settings, allowed overrides, effective settings and fingerprint. User files have `built_in` and `external_label` false. Imports validate JSON, format, version, naming, settings/ranges and fingerprint. A newly introduced setting takes the recorded base value or nearest installed ancestor's value, with an import disclosure.

11.2 Execution and retained results

Expanded multi-factor gage designs fit explicit variance-component indicator matrices by REML using the pinned Powell optimizer. The classic crossed branch preserves the existing crossed estimator. Study worksheets use `numpy-pcg64-operator-block-shuffle-v1` for randomized order and retain its fingerprint; root operation `tabelz.study_worksheet` has no input tables. Capability table cards display at most 200 rows and 16 columns; the retained semantic result preserves its indices, counts, settings, variance view and plot. Display limits do not redefine the analysis population.

11.3 Historical statistical decisions and printed comparisons

The decision identifiers below preserve the existing specification's cross-references. They are not new architectural decisions.

11.3.1 Decisions

	Decision
Q1	Misprints and conflicts: follow the evident intent and log a deviation.
Q3	Chapter IV Sections E and H are included as optional methods.
Q4	The spread note is the Quick Guide, p. iv.

	Decision
Q5	ndc is truncated with a minimum of 1; the constant is 1.41.
Q6	Range method: Appendix C's 1.19105, not the rounded 1.19.
Q7	K1, K2, K3 computed and rounded to 4 decimals; K1 uses d_2.
Q8	Negative PV^2: boundary PV = 0, ndc = 1, incompatible external TV flagged, guidance withheld; Appendix E boundary labelled truncated.
Q9	Out-of-control ranges are flagged with the instruction; results are still computed. Discarding is not offered.
Q10	The decision basis follows the intended use.
Q11	Unbalanced data are refused, naming the cells; REML only under HypoNum extended.
Q12	Gage R study with the sample SD; MR/d_2^* deferred.
Q13	Cp impact from observed Cp only, on both bases.
Q14	Interaction α 0.25 by default, adjustable.
Q15	Constants beyond Appendix C by the same formula.
Q16	Full precision; display at the report form's precision.
Q17	D_4 3.27 and 2.58 as printed; 3-decimal standard factors otherwise; $D_3 = \max(0, 1 - 3d_3/d_2)$.
Q18	Section E: Chapter II's ndc ≥ 5 as the decision, Section E's meanings as the explanation.
Q19	Only the interaction F is reported.
Q20	The example report's verdict is not followed; Chapter II Section D guidance applies.
Q21	Section E constants: its tables for 2–10, Appendix C beyond.
Q22	No confidence limits under this profile; the construction of the printed limits is not established.
Q23	Section H limits from the pooled SD with $c_4(\nu + 1)$.
Q24	Section H reproducibility divides by r; negative values give 0.
Q25	h and k with the descriptive index; no z for k, no normal claims.
Q26	Every ANOVA output explains the interaction decision and the effect of a lower α.

11.3.2 Deviations and clarifications

	Where	HypoNum
D1	Ch. III B, ANOVA ndc	4, by truncation (printed "≈ 5").
D2	Table III-B 10	GRR % contribution 7.8 (7.76), not 7.9.
D3	Table III-B 10 verdict	Chapter II Section D guidance (Q20).
D4	Tables III-B 7 / A 4	Interaction F only (Q19).
D5	Tables A 5 / III-B 8	Withdrawn correction: the raw-data part variance 1.0864466 rounds to 1.086447; the lower printed value comes from rounded mean squares.

	Where	HypoNum
D6	Table III-B 9	Average and Range %AV 20.0 (20.04), not 20.1.
D7	Range example	$d_2^* = 1.19105$; %GRR 75.6, not 75.7.
D8	Bias method	Parked for the bias specification.
D9	Ch. IV H, step 12	Pooled SD with c_4, as in Figure IV-H 1 (Q23).
D10	Ch. IV H, step 14	Reproducibility subtracts s_E^2/r, not $/m$.
D11	Ch. IV H, consistency	No z for k; the h index is descriptive.
D12	Table III-B 9	90% limits not reported (Q22).
D13	Appendix B, Table B 1	Corrected tolerance formula: 0.84 at 90%, not "never".
D14	Appendix C	Computed constants govern ($d_2(15) = 3.47183$; $d_2^*(2,2) = 1.27930$, $d_2^*(2,4) = 1.20620$; $d_2^*(3,3) = 1.76857$, closed form $\sqrt{2/3 + (6+\sqrt{3})/\pi} = 1.7685743$, where Table C 1 prints 1.76858 — a value reproduced by rounding both moments to five decimals first). Printed cells are kept as comparisons.
D15	Appendix D, MR/d_2^*	Deferred; sample SD used.
D16	Figure III-B 16, %EV	17.61 (the form's own 100×0.20188/1.14610 = 17.6145; raw data 17.612), not the printed 17.62.

Other printed values differ from the full-precision results only through intermediate precision; none changes a band, an ndc or a flag, and HypoNum keeps full precision:

- Table A 5 prints the GRR SD 0.302373 and 6σ 1.814238; the raw data give 0.3023715 and 1.8142291. $\sqrt{0.09143} = 0.3023739$ would round to 0.302374, so the printed pair is consistent with the rounded variance and a *truncated* SD (times 6), not with ordinary rounding. This describes the numbers, not the publisher's undocumented procedure.
- The Section E signal-to-noise ratio is 1.698562 at full precision; the printed 1.698 is 12.38/7.289 = 1.698450 from rounded intermediates. 1.41× either unrounded ratio rounds to 2.395, while 1.41 × 1.698 = 2.394; ndc is 2 on every path.
- The range limit 0.8816 is 0.3417 × 2.58 from the rounded average range; at full precision it is 0.8815. Both flag appraiser B on part 4.

11.3.3 Validation evidence

The worked examples are validation cases in the statistical suite, with the manual as a published reference and the printed discrepancies above as convention notes: `STAT-MSA-AIAG-ANOVA-001` (Figure III-B 15, ANOVA, including Minitab's published 22.68% of tolerance), `STAT-MSA-AIAG-AR-001` (Average and Range, raw data, at the precision the rounded intermediates support), `STAT-MSA-AIAG-RANGE-001`, `STAT-MSA-AIAG-WORKSHEET-001`, `STAT-MSA-AIAG-POOLED-001` and `STAT-MSA-AIAG-CP-001`. They show that the documented calculations are implemented; they are not an independent qualification of the profile.

Part III

Build, release and support

CHAPTER 12

Release: packaging and release baseline

This repository currently produces a Python wheel and source distribution. Native Desktop standalone-bundle creation is a separate release step using pinned PyInstaller in an isolated release environment; it is not implied by a wheel build. The standalone step does not itself create an installer: the Windows NSIS adapter wraps its validated candidate in a separate step.

12.1 Repeatable builds

Run the following from a clean checkout with the locked release interpreter, and retain command output together with artifact hashes:

```
uv build --wheel --sdist --out-dir release-out/python
pwsh packaging/windows/build.ps1 -PythonRoot <release-environment-root> -
    OutputDirectory release-out/windows
pwsh packaging/windows/build-installer.ps1 -CandidateDirectory release-out/windows/
    Tabelz -OutputDirectory release-out/installer -MakensisPath <portable-makensis.exe
    > -ProductVersion 0.1.0 -PythonExe <release-environment-python.exe>
```

On an Apple Silicon macOS release runner, use:

```
PYTHON_ROOT=<release-environment-root> packaging/macos/build.sh release-out/macos
```

For a non-building macOS configuration check, set `DRY_RUN=1` before that command. The check stages caller-owned data files and prints the PyInstaller command without installing tools or building an artifact.

Both native scripts bundle the application QML tree at `tabelz/qml`, the notices and inventory at `tabelz/licenses`, and installed `tabelz` distribution metadata. A project PyInstaller hook uses the `pyside6-qmlimportscanner` import closure (QtQuick, Controls, Dialogs, Layouts, and their runtime dependencies). Candidate validation rejects unapproved QML modules and Qt's GPL-only module families, including Virtual Keyboard, Quick 3D, Quick Timeline, and Graphs, as well as the conservatively excluded Charts/Data Visualization libraries.

The root-level `tabelz_desktop.py` is the PyInstaller entrypoint. The Python build also force-includes QML in the wheel. PyInstaller is a release-tool environment dependency and never an end-user runtime dependency. The caller-provided output directory must not exist: packaging creates it and emits `Tabelz/Tabelz.exe` on Windows or `Tabelz.app` on macOS. The pinned PyInstaller release tool requires Python 3.13; the packaging scripts reject other interpreter minors before building.

Only that `Tabelz` directory or `.app` is the release candidate. The output root also contains staging, build, and generated-spec intermediates; those files outside the candidate are not shipped. Before native packaging, both scripts stage `licenses/THIRD_PARTY_NOTICES.md` and `licenses/dependency-inventory.json` beneath the caller-owned output root, and refuse to continue if either is absent. Release assembly must retain the candidate's bundled copies.

12.2 Signing and notarization

The Windows installer adapter validates the candidate Qt audit before calling pinned NSIS 3.12. It emits an unsigned setup executable named for the HypoNum product version, not the compiler version. It installs per-user without elevation at `%LOCALAPPDATA%\Programs\HypoNum`, supports NSIS `/S` silent install and uninstall, writes only HKCU uninstall registration, and on upgrade replaces only marker-owned application files. A non-empty unmarked directory fails closed. Uninstall removes only the application, shortcut, and installer registry keys; it never owns or deletes `.hnum` projects. This local artifact is unsigned and is not clean-machine evidence.

The NSIS installation owns only its HKCU marker and application files; user projects are preserved across installation, upgrade, rollback, and uninstall.

Signing credentials are supplied only by the release runner's hardware-backed or CI secret store. They must never be placed in the repository, an update manifest, a crash artifact, or a build log. The Windows release operator signs the final standalone executable with an organization Authenticode certificate and records the output of:

```
signtool verify /pa /all <artifact>
```

The unsigned NSIS installer is local evidence only; a published installer must be signed separately. The macOS operator signs every nested bundle and the final `.app` with a Developer ID Application identity, submits the signed archive through `notarytool` using a CI keychain profile, staples the result, and records results from:

```
codesign --verify --deep --strict
spctl --assess
```

The local validation record makes no claim of a signed Windows standalone bundle, clean supported-machine test, macOS package, Apple notarization, or Apple Silicon native smoke. Each is a release-gate evidence item.

The Individual-license Ed25519 issuer private key follows the same exclusion boundary and must not enter staging or a packaged candidate. `src/tabelz/licensing/core.py` contains only the matching public verifier. Before publication, smoke-test a trial launch, a key for the candidate machine, rejection of a key for another Installation ID, and the activation-only expired path.

12.3 Upgrade, rollback, uninstall, and projects

Installers must place application files under their platform application directory and must never register `.hnum` documents as installer-owned data. The required native release test creates a project under a user documents directory, upgrades, rolls back only the application binary, uninstalls, and then opens the project with the remaining build. Release notes must state the oldest project schema that can be opened. The new reproduction format is 1.0.0; earlier development-format projects are rejected rather than migrated. This native installer sequence has not yet been performed.

The public local validator proves wheel/sdist contents and a no-developer-tools isolated Python installation only; it does not equal an installer upgrade test. Native installer upgrade/rollback and clean-machine Desktop smoke remain physical release evidence. The validator also runs calibration and capability desktop protocols through the installed wheel, including bundled QML. It rejects developer-path imports, checks that the source baseline remained unchanged during build/test, retains candidate hashes and workflow evidence, and verifies that a saved project is unchanged by wheel uninstall. This is offscreen engineering evidence on the host

machine, not an executed native installer or clean-machine qualification. The platform-specific mouse, keyboard, DPI, accessibility, export, and installer lifecycle execution is the protocol in chapter 13.

The installed-runtime recipe smoke must additionally run both `hyponumreciperun` and an exported `analysis.py` from outside the source checkout, verify that imports resolve to the installed wheel, and compare selected output values and source identity against the fixture. The wheel must contain `tabelz.recipes`, source codecs, operation adapters, and their numerical dependencies. A successful source-tree run does not establish this packaging property.

12.4 LGPL release obligations

The community PySide6 wheels use the LGPL-3.0-only release path; HypoNum remains proprietary and Qt/PySide are dynamically linked. A candidate bundles prominent notices and the GPLv3/LGPLv3 texts, and its QML closure is validated to exclude unapproved modules. Signed native artifacts must not be published until the release owner has the exact Qt/PySide corresponding source under its control or a valid written offer, release-specific installation/replacement information, and compliance review. Source/offer fulfillment items remain external release blockers. Installation and replacement rights are recorded in chapter 16.

12.5 Database driver licences

Database server import (F158) adds three direct runtime dependencies, decided by the owner on 2026-09-30 (issue 150):

- PostgreSQL through `adbc-driver-postgresql` (Apache-2.0). It was chosen over `psycopg` (LGPL-3.0) so that no second LGPL component and no LGPL legal review is needed. Its wheel statically links libpq (PostgreSQL License), OpenSSL (Apache-2.0) and zlib, and vendors further permissively licensed code. Their notices are reproduced in `THIRD_PARTY_NOTICES.md` as the driver distributes them.
- SQL Server through `pyodbc` (MIT) with the ODBC driver the user installs under Microsoft's terms. Microsoft's driver is not bundled or redistributed.
- Remembered passwords through `keyring` (MIT), which uses the operating-system keychain.

The transitive packages (`adbc-driver-manager`, `jaraco.*`, `more-itertools`, `pywin32-ctypes`, `importlib-resources`) are covered by the resolved CycloneDX inventory. A candidate's notice review checks that the driver notices still match the locked wheels.

12.6 Corresponding-source preparation for version 0.1.0

`packaging/qt-source-manifest.json` binds the reviewed Windows executable and installer hashes to the exact PySide6, PySide6 Addons, PySide6 Essentials, shiboken6, and Qt 6.11.2 runtime. It pins the official Qt source archives by filename, byte count, and SHA-256:

- `qt-everywhere-src-6.11.2.tar.xz` (1,019,661,552 bytes);
- `pyside-setup-everywhere-src-6.11.2.tar.xz` (18,053,248 bytes).

Validate the manifest, release environment, and reviewed local artifacts:

```
uv run python tools/prepare_qt_corresponding_source.py `
  --python .tmp/tblz023-release-py313-env/Scripts/python.exe `
  --candidate-executable .tmp/tblz036-pyinstaller-complete-plotting-v4/Tabelz/Tabelz.
    exe `
```

```
  --installer .tmp/tblz036-nsis-complete-plotting-final/Tabelz-0.1.0-unsigned-setup.exe
```

The release owner can download and verify the exact set into explicitly controlled release storage:

```
uv run python tools/prepare_qt_corresponding_source.py --download D:\release-custody\
    tabelz-0.1.0-qt-source
uv run python tools/prepare_qt_corresponding_source.py --verify-directory D:\release-
    custody\tabelz-0.1.0-qt-source
```

The repository manifest deliberately reports `prepared_not_custodied`. The tool proves archive identity only: it cannot assert release-owner custody, publish a source URL, create a written offer, or perform legal/compliance approval. The responsible owner must evidence those steps in the release record before publication.

12.7 Local crash artifacts

`CrashArtifactStore` writes a redacted JSON record to an application-owned local directory. Upload is not implemented, including when a policy object records opt-in. A future uploader needs a separate permission, destination, and privacy review. Table values are never included.

12.8 Signed update manifests

There is no network updater. The unsigned NSIS installer is local validation evidence, not a published release channel. The readiness primitive is a strict detached Ed25519 verifier. A manifest has exactly these fields: `schema_version`, `channel`, `version`, `artifact_name`, `artifact_sha256`, and `signature`. The signature covers canonical JSON of all fields except `signature`; the client's pinned identity is the public-key SHA-256.

Verification rejects unknown fields, bad signatures, a wrong key identity, artifact substitution, channel mismatch, and version downgrade. This baseline accepts only release versions in plain `X.Y.Z` form, so previews cannot accidentally reach the stable channel. Fetching, displaying a permission prompt, and installation may be added only behind this verification boundary and an architect-approved update design.

CHAPTER 13

Release: native acceptance protocol

Protocol `REL-NATIVE-001`, revision 1.0.0. Not yet executed.

This operator handoff covers the remaining Windows x64 and Apple Silicon release checks. Passing automated source or installed-wheel workflows is not equivalent to passing this protocol. Record each platform separately. A missing machine, signature, observer, artifact, or required check remains **not performed**; an offscreen screenshot never converts it to a pass.

13.1 Local Windows checks and clean-machine acceptance

The development Windows machine can test standalone startup, local native interaction, rendering, and application behavior. Run these checks against disposable synthetic projects and a dedicated application-data directory so existing licenses, history, and documents remain untouched. Label such observations **local Windows** and record the exact binary hash; they do not require a VM.

A VM is optional, not a product requirement. A clean physical Windows machine also works. The clean-machine pass checks that installation does not depend on developer runtimes or pre-existing settings. Use a disposable machine or snapshot for upgrade, rollback, and uninstall checks that could affect an existing installation. Until machines are available, retain the candidate and test package and mark those checks, and every unexecuted Mac check, **not performed**.

13.2 Candidate and disposable test machine

Use a clean, supported Windows machine or VM and a native Apple Silicon Mac with no developer checkout, Python, uv, or development environment on the application path. Retain the OS version, CPU architecture, display resolution and scaling, locale, test date, operator, candidate commit, dependency-lock hash, installer/bundle hashes, previous supported release, and evidence directory identity. Preserve a VM snapshot identifier where applicable. Do not use an employee's live documents or overwrite an existing user installation for lifecycle tests.

Build using the repeatable packaging procedure in chapter 12. Release-owner signing and notarization remain required, and signing credentials stay outside source and logs. Record Authenticode verification on Windows, or codesign, Gatekeeper assessment, and stapled notarization verification on macOS. Record bundled dependency notices and corresponding- source fulfillment review separately; automation cannot approve either.

Run automated engineering preparation first, from the matching checkout:

```
uv sync --locked --all-groups
uv run --locked python -m pytest --junitxml=.tmp/release-regression.xml
uv run --locked python tools/run_validation_evidence.py --output-dir .tmp/release-
    workflows
uv run --locked python tools/run_release_validation.py --work-dir C:/
    TabelzReleaseEvidence/candidate-001
```

Choose a new external path for the release validator on each attempt. On macOS, use a new absolute directory outside the repository instead of the Windows path. Transfer the identified native candidate, the workflow archive's synthetic CSVs, expected results, and saved projects to the clean machine. The archive contains no customer dataset. Retain the ZIP and its checksum with the exact candidate.

13.3 Required physical/native checks

For every row retain actual observations, pass/fail/not-performed status, evidence paths, and any deviation. A screenshot alone cannot establish keyboard or screen-reader behavior. Keep recordings and logs free of activation keys and personal data.

ID	Execute	Acceptance
NATIVE-01	Install identified, signed candidate as a normal user	Launch without developer runtimes; expected signature/platform trust; no unexpected elevation
NATIVE-02	First launch; trial and activation paths on dedicated test identities	Correct trial state; correct-machine entitlement accepted; wrong-machine entitlement rejected; expired/invalid state cannot open analysis workspace
NATIVE-03	Repeat the calibration protocol using actual menu, mouse, and keyboard actions	Same prepared records/exclusion population, 36 analyzed Norris pairs, estimates and interpretations as automated references; saved report and HTML/LaTeX export agree
NATIVE-04	Repeat the capability protocol using its explicit normal model and specification settings	Same included population, model checks, capability indices, and interpretation; no silent model substitution
NATIVE-05	Save each study; close application; reopen project; change exclusions; rerun	Persisted data, exclusions, and immutable report retained; stale warning appears after source change; rerun updates live analysis deliberately
NATIVE-06	Keyboard-only import, setup, run, report, and export; close child windows	Focus visible and in reading order; controls named; no keyboard trap; focus returns appropriately
NATIVE-07	Review 1280x720 at 100%, 1440x900 at 100%/150%, 1920x1080 at 100%/150%/200%; minimum/restored/wide windows	Readable text, no clipped actions or disclosures; plot aspect/axes preserved; warnings and full results reachable by scrolling
NATIVE-08	Use Narrator or VoiceOver on scope, setup, result tables, and errors	Meaningful names/roles and findings available without relying on color or hover; observe actual announcements
NATIVE-09	Export HTML, PDF, and LaTeX; open in the appropriate viewer/-compiler	Notes, tables, interpretations, charts, and source-change disclosures retained; no clipped/missing content; preserve output and reviewed pages

NATIVE-10	Save synthetic projects outside installation; hash; upgrade from previous supported release	Application upgrades; projects remain byte-identical until deliberately saved; both studies reopen correctly
NATIVE-11	Roll back application to the stated compatible version	Projects retained; supported schema opens or unsupported schema is rejected clearly without modifying the project
NATIVE-12	Uninstall; compare project hashes; reinstall and reopen	Only app-owned files removed; projects retained and usable; no invented guarantee for unsupported schemas
NATIVE-13	Interrupt a disposable unsaved study; relaunch and follow recovery; restore a backed-up saved project	Last valid saved project retained; offered recovery matches documented behavior; restored study, data, and reports are usable

The numerical acceptance belongs to the two workflow protocols. The native operator must not invent a new tolerance or round away a discrepancy. PDF and native input checks are additional scope, not covered by offscreen runners. Record unavailable viewers, compilers, or assistive technology as not performed.

13.4 Execution record and release decision

Create one editable record per platform with this header and retain all later reruns rather than overwriting failures:

```
Protocol/version:
Candidate commit and lock SHA-256:
Candidate and installer filenames/SHA-256:
OS/build/architecture/locale:
Machine or VM snapshot identifier:
Previous release and supported project schema:
Operator / executed UTC:
Display/scaling combinations actually tested:
DPI/accessibility observations:
Signature/notarization evidence:
Automated evidence ZIP/SHA-256:
Native check ID | actual observation | status | evidence paths | deviation ID
Open failures / not-performed checks:
Independent reviewer / review date / disposition:
```

Do not mark Gate E complete until both supported platforms have the required accepted evidence and the release owner's outstanding approvals. CI's 90-day artifact retention is a convenient transfer period, not the controlled release archive or retention policy.

CHAPTER 14

Release: licensing operations

14.1 Product and customer workflow

HypoNum Desktop starts with a 30-day offline trial. The trial begins on first desktop launch and is tied to the operating-system installation. During the trial, the customer chooses **Help** > **License...** to copy the Installation ID and enter a key. After the trial ends, HypoNum shows only the activation window and does not load the main application until it accepts a valid key.

An Individual license is an offline, machine-bound Ed25519-signed entitlement. The Installation ID is a one-way Base32 digest of the operating system's local machine identity. HypoNum does not display, store in a license key, or transmit the underlying operating-system identifier. A normal computer rename or hardware upgrade does not change the Windows Installation ID. Reinstalling Windows or moving to another computer produces a new ID and requires a replacement key.

The customer workflow is:

1. Open **Help** > **License...** and copy the 32-character Installation ID.
2. Supply that ID with the purchase.
3. The operator issues a license and sends only the `license_key` value from command output.
4. Paste the key into the License window and select **Activate**.

Keys contain only uppercase Base32 letters and numbers and start with `TBLZ1`. Each key signs the customer label, anonymous license ID, Installation ID, Individual plan, issue time, and expiration time. The default validity is exactly 365 days from issuance.

14.2 Issuing, renewing, and rotating keys

The issuer private key must remain outside the repository and every application or installer artifact. The application contains only its public verification key. Losing the private key prevents compatible new licenses and renewals; disclosure lets another party issue authentic keys. Keep an encrypted backup under separate control.

From the repository's release environment, issue the default 365-day key:

```
$issuer = "$HOME\Documents\Tabelz-License-Issuer\issuer-private.pem"
uv run python tools/license_admin.py issue `
  --private-key $issuer `
  --installation-id "CUSTOMER_INSTALLATION_ID" `
  --customer "Customer Name"
```

Use `--expires-onYYYY-MM-DD` when the commercial record specifies a fixed calendar end date. Treat the JSON output as a private fulfillment record: it contains the customer's working key.

For an early renewal, pass the old key to `renew`. The tool extends from the old expiration when it is still in the future, so the customer retains all paid time. If the key has already expired, the new 365 days begin at renewal:

```
uv run python tools/license_admin.py renew `
  --private-key $issuer `
  --license-key "CUSTOMER_CURRENT_LICENSE_KEY"
```

Generate a different issuer identity only for a new product lineage. `keygen` refuses to overwrite an existing file:

```
uv run python tools/license_admin.py keygen --private-key D:\secure\issuer-private.pem
```

Changing the issuer key also requires replacing the pinned public key and its SHA-256 fingerprint in `src/tabelz/licensing/core.py`; old customer licenses will then stop verifying.

14.3 Trial and enforcement behavior

On Windows, trial records are authenticated and encrypted for the current user with DPAPI. HypoNum maintains copies beneath Local AppData and Roaming AppData, requires them to agree on the original start time, advances the latest-seen time, and rejects protected records that fail integrity or machine checks. Moving the clock back by more than five minutes blocks the trial. While HypoNum remains open, it rechecks at least every six hours and at the exact entitlement expiry; the process closes when the entitlement is no longer valid. A valid paid license can replace an expired trial. The same protected clock anchor applies to paid licenses, so clock rollback or a damaged clock record fails closed.

The license signature prevents key forgery, customer or expiry editing, and substitution of another Installation ID. Offline licensing cannot provide instant revocation, prevent somebody from cloning an entire operating-system image, or make a locally patched executable impossible. The application gate raises the cost of bypassing ordinary trial and key checks; code signing and release integrity remain separate controls.

14.4 Bootstrap dependency license inventory

The bootstrap dependency review was recorded on 2026-08-29:

Dependency	License
`hatchling`	MIT
`hypothesis`	MPL-2.0
`duckdb`	MIT
`mypy`	MIT
`numpy`	BSD-3-Clause
`scikit-learn`	BSD-3-Clause
`scipy`	BSD-3-Clause
`statsmodels`	BSD-3-Clause
`pandas`	BSD-3-Clause
`pydantic`	MIT
`PySide6`	LGPL/commercial dual-license
`pytest`	MIT
`pytest-qt`	MIT
`ruff`	MIT

The `licenses` directory holds third-party notices or redistributed license texts required by packaging and release work.

SPSS, Stata and SAS file import (F154, added 2026-09-28) adds `pyreadstat` (Apache-2.0; `>=1.3, <2`, locked at 1.3.6), which bundles the ReadStat C library (MIT). The third-party notices name both and carry ReadStat's MIT text; `Apache-2.0.txt` beside them is the complete Apache license. Release validation fails a wheel whose notices omit pyreadstat or ReadStat or that lacks the Apache text, and the PyInstaller hook `packaging/pyinstaller-hooks/hook-pyreadstat.py` collects pyreadstat's compiled extensions for both platforms. The dependency adds no transitive package beyond `narwhals` (MIT), which was already locked. Only this reader is used for these formats; readers under GPL or AGPL, such as `pyreadr`, are excluded by the interoperability licence rule.

14.5 Open-format reader packaging

The open-format dependency additions recorded on 2026-10-03 are `rdata` (MIT; `>=1.1,<2`, locked at 1.1.0) and `h5py` (BSD-3-Clause; `>=3.16,<4`, locked at 3.16.0). Parquet and Arrow readers use the existing Polars dependency (MIT), and older MATLAB readers use the existing SciPy dependency (BSD-3-Clause). The shipped third-party notices include rdata, xarray (Apache-2.0), h5py and the bundled HDF5, LZF and zlib components. The complete Apache license text remains a separate shipped notice; recording a Python package's license alone does not replace its bundled-library notices.

The PyInstaller hook `packaging/pyinstaller-hooks/hook-rdata.py` collects rdata submodules while excluding tests and copies rdata and xarray package metadata. The hook `packaging/pyinstaller-hooks/hook-h5py.py` collects h5py data, compiled binaries and hidden imports, including its HDF5 runtime libraries. Release inspection checks the open-format notices and rejects an artifact missing required HDF5 attribution. These hooks and notice checks are packaging controls, not a claim that a new frozen build has been qualified on every supported platform. Reader libraries remain confined to the spawned import worker during normal application import handling.

CHAPTER 15

Release: support triage and project recovery

15.1 Triage

Support should follow this sequence and preserve the user's data boundary:

1. Record the HypoNum version, operating-system version, and smallest reproducible workflow. Do not request source data by default.
2. Ask the user to create a local diagnostic ZIP with `hyponumdiagnosticsbundle`. It contains metadata, redacted logs, and receipts only, and never uploads itself.
3. Check receipt operation IDs, status, warnings, and source diagnostics. A damaged or checksum-mismatched embedded source is a package-integrity issue, never a reason for silent replacement from a similarly named file.
4. For worker failures, run `hyponumdoctorworker` and capture its JSON result in a diagnostic log. Reproduction is an explicit user action; do not ask users to run an unreviewed edited export.

15.2 Recovery

1. Keep the original `.hnum` file unchanged. Copy it before attempting recovery.
2. Open the copy. If a recovery snapshot is offered, select **Restore recovery snapshot**, inspect history for stale operations, and save the result under a new name.
3. Inspect package-integrity and implementation-version errors. A missing original external source does not prevent reproduction from a valid embedded snapshot; a corrupt member must not be silently substituted.
4. If opening fails, send the diagnostic bundle and the error class/message. Raw project or source data requires explicit separate consent and a secure transfer process.

Diagnostic ZIP files are local artifacts. Support staff must handle any user-supplied files under the organisation's approved privacy process.

CHAPTER 16

Release: Qt/PySide corresponding source and replacement

HypoNum is proprietary application software. Its community Qt/PySide6 runtime is dynamically linked under the LGPL-3.0-only release path. The release owner must provide the exact corresponding Qt/PySide source under its control or a valid written offer before publication, together with release-specific installation and replacement information and compliance review.

16.1 Installation and replacement information

The Windows candidate is the `Tabelz` directory. Run `Tabelz.exe` from that directory and do not separate it from `_internal`. A compatible replacement Qt/PySide runtime may be installed by replacing the matching Qt/PySide files under `_internal` while preserving directory layout and ABI compatibility. The user may debug a modified replacement runtime; HypoNum imposes no contractual restriction on reverse engineering for that debugging purpose.

For macOS, use the `Tabelz.app` bundle and retain its internal framework layout. A compatible replacement runtime belongs in the corresponding Qt/PySide locations inside the app bundle. Re-signing and notarization may be required after modification; those platform controls do not remove the user's LGPL replacement and debugging rights.

Each release candidate includes the following files and release metadata:

- `tabelz/licenses/THIRD_PARTY_NOTICES.md`;
- the GPLv3 and LGPLv3 texts; and
- the dependency inventory; and
- release metadata needed to identify the candidate and its corresponding source or offer.

Contact details and a source or offer URL are release-specific and must be recorded in the release record. The candidate's QML closure is validated to exclude unapproved modules.

16.2 Version 0.1.0 source manifest

The manifest `packaging/qt-source-manifest.json` binds the reviewed Windows executable and installer hashes to the exact PySide6, PySide6 Addons, PySide6 Essentials, shiboken6, and Qt 6.11.2 runtime. It pins the official Qt source archives by filename, byte count, and SHA-256:

Archive	Byte count
`qt-everywhere-src-6.11.2.tar.xz`	1,019,661,552
`pyside-setup-everywhere-src-6.11.2.tar.xz`	18,053,248

Validate the manifest, release environment, and reviewed local artifacts:

```
uv run python tools/prepare_qt_corresponding_source.py `
  --python .tmp/tblz023-release-py313-env/Scripts/python.exe `
  --candidate-executable .tmp/tblz036-pyinstaller-complete-plotting-v4/Tabelz/Tabelz.
    exe `
  --installer .tmp/tblz036-nsis-complete-plotting-final/Tabelz-0.1.0-unsigned-setup.exe
```

The release owner can download and verify the exact set into explicitly controlled release storage:

```
uv run python tools/prepare_qt_corresponding_source.py --download D:\release-custody\
    tabelz-0.1.0-qt-source
uv run python tools/prepare_qt_corresponding_source.py --verify-directory D:\release-
    custody\tabelz-0.1.0-qt-source
```

The repository manifest deliberately reports `prepared_not_custodied`. A successful tool run proves archive identity only: it is not evidence of release-owner custody, publication, a written offer, or compliance approval. The tool cannot assert custody, publish a source URL, create a written offer, or perform legal/compliance approval. Those steps must be evidenced in the release record by the responsible owner before publication. Signed native artifacts remain blocked until the source or offer, replacement information, and compliance review are complete.

Part IV

Security and validation evidence

CHAPTER 17

Security: threat model and privacy baseline

Status: accepted baseline, updated 2026-09-09. HypoNum is local-first: opening a project is a validation and decoding operation, not an execution surface. Local analysis plugins and the explicitly invoked `hyponumscriptrun` command execute user-controlled Python in separate processes. Repository, remote extension, update, and MCP execution remain unsupported.

Assets include raw rows, selections, credentials and tokens, authored code, projects, caches, receipts, license keys, issuer keys, and signing material. There is no network updater, authentication, scheduler, connector, marketplace, notification, MCP, or cloud repository. Diagnostics remain local unless explicitly moved.

17.1 Assets and trust boundaries

The implemented boundaries, controls, residual risks, and owners are:

Boundary/assets	Abuse case	Current control	Residual risk/owner
Untrusted files and project packages → project reader: workflow, analysis metadata, embedded sources	Traversal; duplicate or case-colliding members; symlink members; zip expansion; malformed JSON	Canonical member names; duplicate rejection; per-member and total byte limits; streaming checksum validation; schema validation	Archive parser denial of service within stated limits; **Medium**, project owner

User-selected SPSS, Stata or SAS file → pyreadstat (ReadStat, native C parser) in the import worker: file bytes, labels, SAS format catalogs	Malformed or hostile file exploiting a parser defect; huge declared row or label counts; hostile text encoding	Parsed only in a spawned import worker (the desktop process never imports pyreadstat); signature and format-version check before parsing; metadata-only discovery; bounded preview window; chunked reads with the `MAX_ANALYSIS_ROWS` refusal; undecodable text refused with the value named; cancellation terminates the worker; files and catalogs are read as data only, never executed	A native parser defect can crash the worker or run code with the user's permissions inside it (the worker is not a sandbox); pyreadstat is pinned by the lock and updated through dependency review; **Medium**, data-runtime owner
Project reader → Desktop/QML: decoded state and display strings	Project content causing Python/QML execution or network activity	`open_project_document` only reads/validates; QML receives controller state, never project QML	QML engine supply-chain defects; **Medium**, Desktop owner
User-selected source → importer/cache: source rows and local paths	Hostile encoding; cancellation leaks; shared-temp disclosure	Bounded import paths; cancellation cleanup; POSIX owner-only cache/worker directories	Windows ACL policy follows creating user and is not asserted as a portable ACL guarantee; **Medium**, data-runtime owner
Controller → spawned worker: operation specification and intermediate artifacts	Forged/malformed messages; artifact traversal; runaway process	Spawn context; versioned JSON-only protocol; request-ID checks; artifact receipt validation; timeout/cancel/cleanup	Trusted local Python environment can still tamper with its own process; **Medium**, executor owner
Operation code generation → exported Python: operation parameters and paths	Secret or credential export; accidental private-path disclosure	Sensitive-key, signed-URL, userinfo, and absolute-path checks; explicit path consent	Generated code is user-controlled after export; **Medium**, workflow owner

Explicit script run → child Python process: exported or edited code and embedded data	Arbitrary local code execution or data disclosure	User must invoke `hyponumscriptrun`; argv paths are passed without a shell; run entitlement is checked before launch	Child process inherits the user's operating-system permissions and is not a sandbox; **High**, script user

User-initiated database server import → SQL worker → PostgreSQL or SQL Server over the network (outbound database connection): the password, credential-free connection settings, the query and its result rows	Password written to a project, receipt, recipe, log or exported script; eavesdropping or a man-in-the-middle on the connection; a query that writes, runs code or reaches outside the database; hostile or oversized server responses; runaway queries	The desktop never connects; only the spawned SQL worker opens the connection, on explicit user action, never on project open, validation or replay; passwords live only in the operating-system keychain (by the user's choice) or in process memory for the session, and are handed only to the worker; settings are credential-free and credential-like keys are refused in settings and provenance; `tabelz.security.redact_text` removes URI userinfo and ODBC `PWD` values, and worker errors are scrubbed of the password; TLS is required by default (`require`, or `verify-full` with certificate verification), and switching it off is an explicit choice recorded with the snapshot; one SELECT/WITH statement with no writing, session or external words and no SQL Server `NEXTVALUEFOR` (a sequence draw survives a rollback); PostgreSQL's read-only transaction setting refuses the database writes it disallows, and SQL Server queries run in a transaction that is always rolled back; statement timeout, row and byte limits; the result is a checksummed Parquet snapshot	`require` encrypts without authenticating the server, so a network attacker able to intercept the connection can impersonate it unless `verify-full` is chosen with the driver's trust roots configured (libpq's root certificate file for PostgreSQL; for the SQL Server ODBC driver, the Windows certificate store on Windows and the OpenSSL trust locations configured for the driver on macOS); neither the text checks nor the server restrictions are an authorization boundary or prove a query free of side effects (a function or extension may act outside the transaction), so a login granted only the intended reads is recommended; driver parsers (libpq, OpenSSL, the user's ODBC driver) run inside the worker, which is not a sandbox; **Medium**, data-runtime owner

Diagnostics/logs → support user: exception text and identifiers	Passwords; tokens; JWTs; URL userinfo; private paths	Table values are not logged; `tabelz.security.redact_text` is the mandatory text-redaction boundary	Future callers must use this boundary; **Medium**, all feature owners
Updates, remote extensions, repository/MCP → application: executable code and credentials	Remote code execution or credential exfiltration	Unsupported and fail-closed; no network client or remote extension loader	Feature requires a signed-update and permission decision recorded in the architecture before implementation; **High**, architect approval required before enabling
Local analysis plugin → active table: arbitrary Python and one explicitly selected table	Arbitrary code execution; broad local access; data disclosure; runaway process; malformed results	Manifest-only discovery; no execution on project open; explicit menu invocation; fresh spawned worker; checksummed temporary Parquet input; strict bounded JSON setup/results; timeout, cancel, stale-response checks; visible trust warning	Worker is not a sandbox and code retains the user's filesystem, process, and network permissions; only trusted plugins may run; **High**, plugin user/author
License issuer → Desktop startup: issuer key, customer entitlement, trial time	Forged/shared key; expiry editing; clock rollback; trial reset	Private Ed25519 key excluded from releases; pinned public key; signed machine and expiry claims; DPAPI-protected mirrored trial state; last-seen clock check; main QML loads only after entitlement evaluation	Privileged local attacker can patch or replace executable, delete all state, or clone an OS image; offline licenses cannot be instantly revoked; **Medium**, release owner

17.2 High-severity disposition

There are no silently accepted high-severity findings in the implemented surface. Remote updates and extensions remain absent and fail closed. Local plugin execution has an unavoidable, visible high residual risk because process isolation is not an operating-system sandbox. It is opt-in trusted-code execution, does not occur while discovering plugins or opening projects, and is not represented as safe for untrusted code.

17.3 Process and network policy

Only local workers, the plugin worker, and narrowly scoped data-materialization workers may create subprocesses; they use Python's spawn context and JSON control messages. Project open, project validation, code generation, plugin discovery, and QML file selection must not execute

plugin Python or make a network request. The only outbound network connection HypoNum itself makes is a read-only database server import (PostgreSQL or SQL Server), opened by the SQL worker on an explicit user action and encrypted by default; opening, validating or replaying a project never reconnects, because a server import is kept as its saved snapshot. HypoNum itself supplies no network capability to a plugin, but trusted plugin code can use every capability already available to the user's Python process. A run receives only the active table from HypoNum; this data-scope boundary does not restrict what arbitrary Python can read from the operating system.

17.4 Open-format reader boundary

The open-format import implementation, reviewed here on 2026-10-03, dispatches Parquet, Arrow IPC/Feather version 2, R data files, MATLAB files and HDF5 files to fixed readers in spawned data-materialization workers. Reader libraries are loaded inside those workers, not the presentation process. Files are decoded as data; importing them does not launch R or MATLAB or execute their programs. Unsupported objects and formats are refused rather than silently converted to another interpretation. Imports retain an owned snapshot and source provenance; subsequent project replay does not depend on executing the original source file.

Object discovery is capped at 1,000 objects, and selected previews use bounded row and column windows. Materialization checks the analysis row limit and workspace budget. Oversized returned windows are reduced, and an individual row exceeding the budget is refused. R and older MATLAB decoders can still materialize a whole object internally: their pre-decoding input checks use a 512 MiB limit, with streamed decompressed-byte counting for compressed R inputs and file/shape estimates for older MATLAB inputs. Neither that limit nor the output workspace budget is a bound on decoder working memory.

The HDF5 table reader follows hard links within the selected file only. It does not follow external links and refuses external-storage and virtual datasets. Cancellation terminates the reader worker. These are application controls, not an operating-system sandbox: a vulnerable native decoder can still crash or exercise the user's process privileges. Worker separation must not be represented as making an untrusted file safe or as imposing a hard operating-system memory cap.

17.5 License-key custody

The Individual license issuer private key is release-owner secret material. It must never be stored in this repository, application data, diagnostics, build staging, release artifacts, or command logs. Release binaries pin only the corresponding public key and fingerprint. Customer keys are bearer secrets bound to an anonymous Installation ID; support and fulfillment records must treat them as private. License verification and trial evaluation are local and make no network request. Cryptographic signatures prevent a modified claim from verifying, but do not make local executable patching impossible.

17.6 Reproducible validation

Run the deterministic local record without network access:

```
uv run python tools/run_security_validation.py --offline --output .tmp/security-
    validation.json
```

The output contains a lockfile dependency inventory, installed-metadata license inventory, a CycloneDX 1.5 SBOM beside it, and a secret-scan result without printing matched values. It

refuses to overwrite existing output or SBOM files. The narrowly scoped scanner allowlist contains only exact fixture file/line locations; a new value in that file is still reported. It never contains production paths. The timestamped report is evidence, not byte-deterministic output or a project fingerprint.

Vulnerability status is truthfully `offline_not_run` until a scanner has run. For a connected, reviewable scan, use the pinned command recorded in that record:

```
uvx --from pip-audit==2.10.1 pip-audit --strict
```

Capture the scan's tool, database, version, and time metadata beside the generated artifact, never in deterministic project fingerprints. Security validation records retain revision, lock, platform, tool versions, fixture hashes, results, and limitations; no formal certification is claimed.

CHAPTER 18

Validation: development checks and integration

Plan/Policy. This chapter defines validation scope during development. It does not record a passing execution or change statistical acceptance criteria. `AGENTS.md` gives agents the operational rules; this chapter explains their application. Independent qualification remains governed by Chapter 19.

18.1 Assign checks to the change

Documentation authorship and handoff. The project owner explicitly designates the statistician, who may be a human or an AI model. Only that statistician authors and approves the canonical user manual and statistical validation reports, including their presentation and supporting report content. An implementation agent supplies a separate draft and reproducible evidence in its handoff; a feature assignment does not grant permission to edit those documents. Reading and building unchanged sources, and transporting an unchanged approved patch, do not transfer authorship. Content-changing conflict resolution returns to the statistician.

The draft identifies implemented behavior, actual menu paths, defaults, assumptions, units, formulas, sources, reproducible examples, observed results, limitations and retained checks. Proposed behavior and unresolved discrepancies remain explicit. Raw test evidence may be maintained by its implementation owner; report conclusions require statistical review. The customer report groups evidence by analysis family and excludes internal development history. Its assessed scope and product version must be clear. The manual adds practical workflows and instruction without treating an unassessed method as validated.

Before implementation, identify affected behavior, exact test files or node IDs, and who owns broader regression. Directory names do not establish test cost: `tests/unit/` also contains subprocess, controller, recovery, benchmark and offscreen QML checks. The default pytest configuration selects all of `tests/`, including integration and validation tests.

Ordinary feature implementation and merge. Run functional tests for the new behavior and affected callers. Numerical work needs a small justified correctness example or mathematical invariant, relevant boundary/failure cases and deterministic settings where applicable. Changes to a workflow also need relevant identity, exclusion, persistence/replay and request lifecycle checks. Table mutations need lineage and undo/redo coverage. Reuse existing guarantees and add tests for new behavior or changed integration; duplicating the same complete workflow at every layer is unnecessary. For a feature exposed through the GUI, include one test through the real menu/setup, request, engine and results path. Check parameter forwarding, displayed values and labels, rather than just window construction or a mocked engine. Successful execution alone does not establish numerical correctness. Focused acceptance is sufficient for delivery and merge of an ordinary isolated feature; adding a model or menu item does not automatically require the full unit, desktop, integration or application suite.

Affected existing behavior. Calling an unchanged helper does not require rerunning every test of that helper or its other callers. Changing helper behavior requires tests of affected callers. Choose neighboring checks from the actual behavior changes, not merely imports or touched filenames. Adding controller, bridge or menu forwarding and registration alone does not require all desktop tests or constitute a broad refactor. Test the specific shared interaction affected.

Shared foundations and integration checkpoints. Changes to shared state transitions, execution, persistence, row identity or other behavior used across many features warrant broader checks appropriate to their impact. Explain that impact before expanding scope. Full application regression belongs to planned integration checkpoints, release candidates or changes whose scope justifies it, rather than every feature merge. One integrator owns that run on the combined source baseline; authors do not each repeat it. Existing CI still selects the full suite automatically, but that configuration does not require duplicating it locally during feature work. Executed CI evidence applies only to its identified baseline; configuration alone, an earlier commit or different uncommitted contents cannot establish a pass.

Broad refactor. Run the full unit suite and affected integration, persistence, controller and QML checks on the combined result. An implementer acting as integrator owns this obligation. Preserve compatibility contracts and existing acceptance thresholds.

Independent validation and release. Retained independent-reference comparisons, extensive simulation studies, native platform and display/accessibility matrices, and packaged execution protocols have separate owners and tasks. Deferring these from feature delivery does not defer basic correctness, justify ignoring a numerical discrepancy or establish that a new feature is statistically qualified.

Run applicable static checks as well. Documentation-only work requires the changed corpus build and any affected documentation/policy checks, rather than numerical or GUI regression. Public contract changes require schema checks and their compatibility coverage. Validation status must distinguish a feature's focused acceptance, combined regression and independent qualification.

18.2 Bound repetition and investigate slow runs

Use explicit pytest paths or node IDs, `--maxfail=1` and `--durations=10` during implementation. Fail-fast runs are incomplete when they encounter a failure; they are not evidence that remaining tests passed. Expensive runs should retain logs and JUnit under a fresh `.tmp/artifacts/` path, identified by their source baseline, environment, command and result. Retain failed evidence too.

When the planned acceptance checks pass, stop unless another edit, failure or specific unresolved risk changes the verification need. After a fix, rerun the failed case and affected checks first. A later focused pass does not relabel an earlier full-suite failure as a passing full suite. A final combined pass is still required where full regression is the acceptance gate.

Ten minutes of local validation triggers a diagnostic checkpoint, not an automatic cutoff or a waiver. Inspect progress, test timings, subprocess startup, worker cleanup and competing jobs before adding more runs. Do not run multiple expensive suites concurrently on the same host. A timeout is a failed check to investigate; increasing timeouts or sleeps simply to make it pass is not a correction.

Investigate apparently unrelated failures with the specific test on the task's unchanged base in an isolated `.tmp/worktrees/` checkout, using the same locked environment. Do not reset the feature tree. A matching baseline failure supports reporting a pre-existing defect, but does not make the combined suite green or excuse an affected requirement. If required acceptance remains unresolved, report its exact blocker and keep it pending. Separate unrelated repair work instead of repeatedly expanding the feature's test scope.

18.3 Improve suite runtime without reducing assurance

Use retained timings to select the next optimization. Distinguish repeated unnecessary suites, genuinely costly tests and failures caused by lifecycle or resource contention. Optimize the responsible fixtures, redundant startup or cleanup only after reproducing the cause. Small deterministic functional fixtures can complement large reference/performance protocols; they do not replace those release obligations. Any future markers, sharding or parallel execution must preserve the full integration/release inventory and account for Qt ownership, spawned workers and native numerical thread pools. No such runner or test optimization is established by this policy change.

CHAPTER 19

Validation: regulated-use master plan

Status: proposed engineering and quality handoff; not an executed validation report.

Prepared: 2026-09-11.

Audience: software engineers, statistical reviewers, validation/quality leads, release engineers, and pilot customers.

19.1 Executive summary

Validate a specific HypoNum release for explicitly defined uses, methods, configurations, data limits, and operating environments. Do not claim that the entire application is universally ”validated,” ”ISO compliant,” or ”NIST certified.”

The program has two deliverables:

1. A supplier release-validation package demonstrating numerical correctness, workflow integrity, and release reliability.
2. A customer qualification package for establishing suitability in the customer's actual controlled environment and intended workflows.

NIST reference datasets are an important starting point, but they do not cover all HypoNum methods or establish software certification or traceability to NIST. See the NIST StRD FAQ.

This is a staged implementation plan, not an instruction to build every control immediately. First approve scope, assess gaps, and select a bounded first release. Regulatory applicability and statistical acceptance criteria require qualified human review.

19.2 Start here: first five working days

[ ] Appoint a validation/quality owner and an independent statistical reviewer.
[ ] Select the first industry, intended workflows, and supported deployment environments.
[ ] Inventory current features and freeze the candidate source/dependency baseline.
[ ] Create requirements and risk registers with stable IDs.
[ ] Approve the reference-data format, comparison policy, and evidence format.
[ ] Archive initial NIST univariate, regression, and ANOVA reference cases.
[ ] Implement the first independent numerical comparisons and harness self-tests.
[ ] Convert previously confirmed defects into traceable workflow regression cases.
[ ] Produce a gap assessment for regulated-record controls.

First milestone: one fully evidenced workflow from import through preparation/exclusion, analysis, reporting, and save/reopen, plus an initial NIST reference suite. Then extend the same approach across the approved method inventory.

19.3 1. Scope, ownership, and claims

19.3.1 Working assumption

HypoNum supports industrial statistical analysis and quality decisions. This plan does not assume that HypoNum is a clinical diagnostic device, a medical-device software function, or an autonomous batch-release system. Such uses require separate applicability and risk assessments.

The intended-use statement must identify:

- Users, industry, jurisdictions, and decisions supported.
- Included analytical functions and method configurations.
- Supported platforms, hardware, locales, input formats, and data sizes.
- Statistical assumptions and supported numerical ranges.
- Whether HypoNum creates, modifies, retains, or transmits regulated records.
- Controls supplied by HypoNum versus customer infrastructure and procedures.
- Explicit exclusions, including unvalidated plugins and experimental methods.

"Calculate process capability to support manufacturing quality review" is a useful starting intended use. "Perform statistical analysis" is too broad.

Each feature/configuration must have an explicit status: proposed, in validation, validated for a named release/scope, or excluded with rationale. New features do not inherit validation from earlier bug-fix verification.

19.3.2 Roles

Role	Responsibility
Product/process owner	Approves intended use, business workflows, scope, and limitations.
Validation/quality lead	Owns risk assessment, protocols, traceability, deviations, evidence control, and release approval.
Software/test engineers	Implement the harness, tests, controls, and defect corrections.
Independent statistician	Reviews method definitions, reference results, tolerances, discrepancies, and numerical conclusions.
Release engineer	Produces and qualifies identified binaries on supported clean environments.
Pilot customer/process representative	Demonstrates actual intended workflows and customer-side controls.

The implementer must not be the sole approver of high-risk expected results or final validation conclusions.

19.3.3 Select the applicable framework

Intended use	Starting framework	Required planning action
Medical-device production/quality systems	FDA February 2026 Computer Software Assurance guidance; ISO/TR 80002-2:2017	Assess intended use and process risk; identify assurance activities and records.
Pharmaceutical GMP workflows	Applicable GMP requirements, EU Annex 11, and FDA electronic-record requirements where applicable	Map lifecycle validation, data integrity, security, retention, and operational responsibilities.
Testing/calibration laboratories	ISO/IEC 17025:2017	Identify requirements relevant to laboratory methods, software, data handling, and reporting.
Automotive/other industrial quality	Applicable customer requirements and selected statistical standards/manuals	Validate the exact methods, editions, and configurations required.

Sources: FDA CSA guidance, ISO/TR 80002-2, EU Annex 11, ISO/IEC 17025.

These frameworks are not interchangeable certifications. For example, Part 11 applicability depends on the records and actual use, not simply on selling to an FDA-regulated organization. Have a qualified reviewer document applicability using the FDA Part 11 scope guidance.

Reconfirm publication, amendment, and effective-date status when approving the validation baseline. Draft requirements must not be represented as effective requirements.

19.4 2. Engineering work packages

Planning estimate: approximately 12-16 weeks for a bounded first release with two engineers, an independent statistician, and a regularly available validation/quality lead. This is not an estimate for qualifying every possible method or implementing major missing record-management capabilities. Material gaps extend the schedule.

19.4.1 VAL-01: Approve scope and freeze the baseline

Owner: product owner and quality lead. Target: weeks 1-2.

Tasks:

- Inventory every user-facing method, configuration, transformation, export, and plugin boundary.
- Identify the first workflows and feature subset to qualify.
- Record source commit, dependency lock, supported platform matrix, and candidate build process.
- Document exclusions, input limits, and customer responsibilities.

Deliverables: approved validation plan, intended-use statement, feature/configuration matrix, baseline manifest.

Acceptance: every candidate feature has an explicit scope decision; the baseline can be reconstructed.

19.4.2 VAL-02: Establish requirements, risk, and traceability

Owner: quality lead and statistician. Target: weeks 1-3. Depends on: VAL-01 scope.

Tasks:

- Assign stable requirement and risk IDs.
- Assess failures by their effect on data integrity, product quality, safety, and user decisions.
- Specify method behavior and acceptance criteria before qualification execution.
- Link each requirement to risks, controls, test cases, and retained evidence.

Minimum traceability fields:

```
requirement_id, intended_use, method/configuration, risk_id, risk_rationale,
control, test_ids, acceptance_policy, evidence_ids, status, approver
```

Acceptance: every in-scope requirement has planned evidence and an accountable owner. Counts and code coverage alone are not a substitute for risk coverage.

19.4.3 VAL-03: Build the reference library and validation harness

Owner: validation engineer and statistician. Target: weeks 2-4. Depends on: approved case/-comparison specifications.

Tasks:

- Archive original reference inputs, certified or independently derived expected outputs, provenance, and checksums.
- Implement checked input adapters and output-specific comparisons.
- Produce machine-readable results and a human-readable evidence summary.
- Add harness self-tests proving that altered inputs, expectations, and settings fail appropriately.
- Require review of all reference and tolerance changes; prohibit automatic rebaselining.

Acceptance: one numerical case and one complete workflow execute reproducibly; deliberate faults are detected; expected results have independent approval.

19.4.4 VAL-04: Qualify statistical methods

Owner: engineers and independent statistician. Target: weeks 3-8. Depends on: VAL-02 and VAL-03.

Tasks:

- Execute applicable NIST cases and non-NIST method-specific references.
- Cover normal inputs, numerical boundaries, invalid inputs, and decision boundaries.
- Add mathematical property tests and simulation evidence where justified.
- Investigate every discrepancy against the approved method definition.
- Correct defects or formally narrow the supported domain; do not hide failures by changing expected values.

Acceptance: every in-scope method/configuration meets its approved criteria; unresolved discrepancies prevent a validated claim for that scope.

19.4.5 VAL-05: Qualify workflows and data integrity

Owner: test engineer. Target: weeks 4-9. Depends on: harness and relevant method tests.

Tasks:

- Execute import-to-report protocols through actual product paths.
- Test row identity, transformations, exclusions, concurrency, reporting, persistence, and recovery.
- Exercise actual GUI actions as well as controller and engine tests.
- Maintain a permanent regression case for every previously confirmed defect.

Acceptance: the displayed/exported result describes the correct data and settings, survives supported persistence operations, and cannot be silently replaced by stale work.

19.4.6 VAL-06: Resolve regulated-use control gaps

Owner: engineering and quality. Target: weeks 3-10 or longer. Depends on: applicability/risk assessment.

Tasks:

- Assess auditability, access, original-data preservation, retention, backup, approvals, and plugin restrictions.
- Allocate each required control to HypoNum, customer infrastructure, or an integrated record system.
- Implement and verify necessary controls, or restrict the intended use.
- Document and test customer-side responsibilities; do not assume an export alone closes record-integrity requirements.

Acceptance: no applicable requirement is left as an unsupported compliance claim or a paperwork-only substitute for missing behavior.

19.4.7 VAL-07: Qualify release binaries

Owner: release engineer and quality. Target: weeks 10-12. Depends on: passing in-scope functional tests and required controls.

Tasks:

- Freeze and identify the actual customer installers and binaries.
- Execute approved protocols on clean supported machines, including native Apple Silicon if claimed.
- Test install, upgrade, uninstall, project preservation, failure handling, recovery, and supported locales.
- Retain exact environment and build evidence.

Acceptance: all claimed environments have executed evidence; extrapolations and unavailable environments remain explicitly identified, not marked passed.

19.4.8 VAL-08: Independent review and customer qualification package

Owner: independent reviewers and pilot customer. Target: weeks 12-16. Depends on: VAL-07 and resolved deviations.

Tasks:

- Review traceability, raw evidence, discrepancies, limitations, and release identity.
- Execute representative customer workflows and customer-side controls.
- Approve the validation summary and ship the qualification package.

Acceptance: statistical and quality approval covers the exact release and scope; customer qualification responsibilities are explicit.

19.5 3. Statistical reference strategy

19.5.1 Existing HypoNum foundations

Reuse existing capabilities rather than creating an unrelated validation stack:

- `src/tabelz/analysis/goldens.py`: golden artifact verification and comparison.
- `src/tabelz/analysis/profiles.py`: versioned method-profile structures.
- `src/tabelz/provenance/`: analysis and environment provenance.
- `tools/run_reliability_validation.py`: recovery and failure-boundary evidence.
- `tools/run_release_validation.py`: release-artifact checks.
- `docs/engineering/validation/reliability.tex`: existing evidence scope and limitations.

Verify their current interfaces before integration. Existing scaffolding does not establish that all current features already participate in it.

A saved HypoNum output is regression evidence: it can detect change. It becomes correctness evidence only when its expected values have independent justification. Likewise, comparing a wrapper against the same dependency function it calls is useful integration testing, not independent algorithm validation.

19.5.2 NIST reference collections

Collection	Initial cases	Outputs to compare
Univariate statistics	NumAcc1-4, then remaining applicable cases	Mean, standard deviation, and every other supported certified output.
Linear regression	Norris, Pontius, Longley, Wampler1-5; other cases where the exact model is supported	Coefficients, standard errors, residual variation, and other supported certified outputs.
Balanced one-way ANOVA	SiRstv, AtmWtAg, SmLs01-09	Sums of squares, mean squares, F-statistics, and other supported certified outputs.

Sources: NIST univariate datasets, linear regression datasets, ANOVA datasets.

For every reference case:

1. Retain the original dataset and reference results, source URL, retrieval date, license/usage information, and checksum.

2. Preserve reference decimal strings so parsing does not silently reduce precision.
3. Independently verify row count, column mapping, model, settings, and input interpretation.
4. Exercise the implementation used by HypoNum, not a test-only replacement calculation.
5. Compare all supported reference outputs and retain discrepancies.
6. Record unsupported models explicitly; do not silently substitute a different model.

Use all applicable difficulty levels. Passing easy cases is not evidence of robustness on ill-conditioned data. Passing NIST cases is also not evidence that every customer dataset is handled correctly.

19.5.3 Methods outside NIST coverage

Area	Reference approach	Critical cases
Hypothesis tests	Independently derived examples and separately implemented statistical methods	One/two-sided tests, alpha, confidence intervals, multiplicity, ties, exact/asymptotic options.
SPC	Selected chart definitions and independently calculated sequences	Every point and limit, subgroup sizes, violation rules, Phase I/II behavior where supported.
Capability	Exact approved method definitions and independent calculations	Within versus overall variation, one-sided specifications, nonnormal models, transformations, tails, intervals.
Gage studies/attribute agreement	Independent calculations and applicable authorized examples	Crossed/nested models, pooling, zero variance, interactions, bias, weights, denominators.
DOE	Hand-verifiable designs and independent model fits	Randomization, blocks, aliases, rank, error terms, pure error, lack of fit, optimization constraints.
Logistic regression/distribution fitting	Independent implementations and constructed examples	Coding, likelihood, estimates, intervals, convergence, separation, parameter boundaries.
Power/sample size	Independent calculations and simulation	Tail choices, effect conventions, allocation, rounding, achieved power at recommended sample size.
Plots/Pareto/exploratory methods	Exact small datasets and method-specific cases	Counts, ordering, percentages, omissions, sampling disclosure, labels, units, interpretation.

Independence must be assessed: two packages can share the same algorithms or numerical dependencies. Combine certified/derived references, independent implementations, and mathematical properties rather than assuming any second package is an oracle.

19.5.4 Standard-based methods

Select standards according to intended use; do not implement every standard merely because it is listed here:

- Shewhart charts: ISO 7870-2:2023.
- Capability/performance: applicable ISO 22514 parts; Part 2:2026 is a current starting reference.
- Measurement-process capability: ISO 22514-7:2021, including applicable amendments.
- Interlaboratory precision, if in scope: ISO 5725-2:2025.
- Automotive measurement-system methods, if claimed: AIAG MSA manual.

Acquire authorized copies and create a clause-to-method-to-test mapping. Public abstracts identify relevance but are not enough to establish implementation conformity. Do not redistribute licensed text or datasets without permission.

These methods are not interchangeable: interlaboratory precision, crossed Gage R&R, and measurement-process capability have different scopes and assumptions.

19.5.5 Required method specification

Before comparing outputs, approve a short specification for each method/configuration:

- Equations, estimators, assumptions, and valid input domain.
- Missing values, exclusions, category ordering, and weighting.
- Degrees-of-freedom and variance conventions.
- Confidence intervals, alternatives, and multiplicity adjustments.
- Pooling, truncation, boundary rules, and default settings.
- Convergence and explicit failure/warning behavior.
- Reference edition/location and versioned method-profile identity.

Resolve specification differences before comparing packages. Never select a reference merely because it agrees with Tabelz.

19.6 4. Acceptance and comparison policy

Use output-specific acceptance rules, approved before evaluating qualification results:

- Exact comparisons for counts, category labels, selected populations, and discrete decisions.
- Justified absolute/relative tolerances for continuous numerical outputs.
- Appropriate tail-probability comparisons; a loose absolute tolerance must not equate zero with a meaningful small probability.
- Explicit expectations for undefined results, rejected inputs, convergence failures, and warnings.
- Tests below, at, and above decision thresholds.
- Separate numerical correctness from display rounding and formatting correctness.

The existing general golden tolerance is not automatically the validation tolerance for every output. Difficult reference cases may expose a limitation requiring correction or a justified supported-domain restriction, not an unreviewed tolerance increase.

For methods without adequate closed-form references, use simulation studies of confidence-interval coverage, false-positive rates, or achieved power as appropriate. Predefine sample sizes,

seeds, Monte Carlo uncertainty, and acceptance rules. Do not rerun simulations repeatedly until one passes.

Add mathematical property tests where valid: invariance to suitable shifts/scaling or category relabeling, equivalent data representations, and analytically known limiting cases. Do not assert row-order invariance for methods where order is meaningful, such as control charts.

19.7 5. Implemented harness and evidence contract

The statistical validation suite is implemented. It runs on demand before a major release only; it never runs in CI or in the normal test suite, whose harness self-tests (`tests/validation/test\_statistical\_*.py`) check the harness itself, not HypoNum statistics.

Layout

```
src/tabelz/validation/statistical/
  schema.py       case (v2) and expected-file (v2) contracts, comparison kinds
  bindings/       one binding per entry point: strict, call, executor,
                  operation or in_process, exactly as the desktop calls it
  runner.py       runs cases through the single analysis engine
  comparisons.py  tolerance, tail_probability, exact, sequence, set, absent,
                  contains and interval comparisons
  inventory.py    menu items and entry points read from the code
  coverage.py     entry-point coverage and the ratchet
  simulation.py   pre-registered simulation studies (in process)
  wiring.py       GUI wiring evidence runner
  release.py      the release rehearsal and its row-free evidence summary

tests/validation/
  reference_data/
    statistical_suite.json  index of cases with SHA-256 hashes
    cases/<family>/         one case file per case
    expected/<family>/      one expected file per case
    datasets/               CSV bytes plus a provenance record each
    discrepancies.json      every failed output, categorized
    inventory/              menu_items, entry_points, map, coverage baseline,
                            wiring
    simulation/             pre-registered simulation specifications
  generators/r/             R generators (renv lockfile) for expected files
  wiring/                   GUI wiring tests, skipped unless BEYONDSTAT_VALIDATION=1

tools/run_statistical_validation.py
  run        indexed cases (all, --family, --case or --smoke)
  inventory  regenerate or --check the menu and entry-point inventory
  coverage   check the ratchet; --release makes it a hard gate
  simulate   run pre-registered simulation studies (--all or --sim)
  wiring     run every menu item's GUI wiring tests, one process each
  release    inventory --check, coverage --release, run, simulate --all
             and wiring in order; --evidence writes the row-free summary
```

Cases, expectations and tiers

A case (schema 2.0.0) names its family, entry point, menu items, case kind, dataset path and hash, excluded rows, configuration, expected-file path and hash, and its oracle: `certified` (NIST),

`published`, `independent\_implementation` (R 4.6.1 with pinned packages) or `analytic`, with the generator path and convention notes. An expected file lists output paths with one comparison each, deferred outputs with a reason, or an expected error class and message fragment. Closed-form results use a relative tolerance of 10^{-9} and iterative fits 10^{-6}; counts and labels compare exactly. Every failed output is appended to `discrepancies.json` and must be categorized as a suspected HypoNum defect, a suspected reference error, a convention the manual leaves undefined, or reference precision below the tier; an expected value or tolerance is never changed after a failure is seen.

Coverage ratchet

`inventory/map.json` traces every menu item to the entry points it reaches. Coverage fails when a mapped entry point has neither a case nor an entry in `coverage\_baseline.json`; the baseline may only shrink. Under `release` any remaining baseline entry or unresolved trace fails the stage.

Simulation and wiring evidence

Simulation studies check coverage, Type I error and achieved power where no closed form exists. Each specification is committed before its first run; the acceptance band is nominal $\pm 3\sqrt{\text{nominal}(1-\text{nominal})/R}$; a study outside its band is recorded, never rerun with a new seed. Wiring evidence proves that each statistical menu item reaches its entry point with the chosen settings and displays what the engine returned; new wiring tests wrap, never replace, the engine transport.

Keep original fixtures separate from normalized inputs and expected results. Store bulky executed evidence in controlled release storage, linked to the source baseline and manifest; do not rely on transient CI logs as the only record.

Runner requirements:

- Machine-readable results plus a human-readable summary.
- Expected and actual values, error measurements, acceptance rules, and outcome.
- Requirement/risk/test/evidence traceability.
- Source commit, dependency lock identity, installer hash where relevant, OS/architecture, and relevant numerical-library configuration.
- Execution time, case/reference versions, and review/approval references.
- Explicit passed, failed, blocked, and excluded statuses with rationale.
- A failing exit status when required evidence fails or is missing; no silent qualification skips.
- Offline replay from archived references and pinned environments where supported.

Harness self-tests must deliberately alter expected results, inputs, column mapping, and settings and demonstrate detection. Use a controlled change/review process for fixture updates. Never generate authoritative expected results by copying the implementation under test without independent verification.

19.8 6. End-to-end workflow protocols

Stage	Required evidence
Import	Exact values/types, row counts, missing-value interpretation, source identity, supported units/metadata.
Prepare	Correct formulas, imputation, joins, grouping, sorting, deletion, and exclusion effects.
Analyze	Correct dataset, included population, configuration, and numerical result.
Concurrent work	Late results cannot populate another dataset or replace a newer analysis.
Save/reopen	Data, settings, formulas, exclusions, reports, and provenance retain meaning.
Report/export	Values, labels, units, limits, confidence levels, and truncation disclosures agree with canonical results.
Recover	Interrupted saves preserve the last valid project; restored data/results are checked.
Migrate	Older projects retain meaning, or controlled reanalysis is explicitly required.

Run important protocols through actual GUI actions as well as engine/controller tests. Verify user-selected settings reach the analysis, prompts prevent unintended loss, and exports represent the intended result.

19.8.1 Permanent regressions from previously confirmed defects

- [ ] Repeated saves of modified imported data preserve import/snapshot semantics.
- [ ] Same-count swaps of excluded rows change report/source interpretation correctly.
- [ ] Sorting excluded rows before save preserves exclusions after reopen.
- [ ] Fresh reports remain current after save/reopen; genuine changes mark snapshots stale.
- [ ] Dataset switches and cell edits suppress late or invalid analysis results.
- [ ] Cache prefetch cannot restore pre-edit values.
- [ ] Non-default significance levels propagate to multiple-comparison output.
- [ ] Integer arithmetic does not silently overflow; fractional imputation is not truncated.
- [ ] Zero-repeatability Gage R&R retains appropriate interactions.
- [ ] Block-aware DOE diagnostics use correct pure-error groups.
- [ ] Large-offset regression remains numerically accurate or reports a supported-domain limitation.
- [ ] CSV/chart/report exports preserve labels, units, limits, paths, and disclosures.
- [ ] Interrupted writes, worker timeouts, and recovery preserve data and terminate work appropriately.
- [ ] Credentials are excluded from diagnostics under the documented redaction policy.

Extend this list from the defect register; it is not a replacement for requirements-based coverage.

19.9 7. Regulated-record control assessment

Assess applicability and responsibility for:

- Attributable records of who changed what, when, and why.
- Preserved original data and reviewable transformations/exclusions.
- Controlled access and authorization.
- Protected audit trails and review procedures.
- Retention, retrieval, backup, and demonstrated restoration.
- Approval and electronic signatures where applicable.
- Reliable identification of software, methods, input data, population, and configuration.
- Restrictions on plugins and unapproved methods.

Checksums and editable histories alone do not establish a protected audit trail. Exporting a PDF is not automatically an adequate substitute for preserving required electronic data/history.

For each applicable control, specify its owner and location: HypoNum, customer infrastructure, or an integrated controlled system. Validate the complete arrangement. If a required capability is absent, implement it or restrict the validated intended use.

EU Annex 11 addresses lifecycle traceability, test adequacy, migration integrity, security, backup, audit trails, and periodic evaluation. See the official Annex 11 text. This plan does not assert current HypoNum conformity.

19.10 8. Release qualification and approval

Qualify the actual customer binaries on clean supported environments, not just a developer checkout.

Include installation, launch, upgrade, uninstall, project preservation, approved locales, numeric/date parsing, path handling, resource limits, permission failures, disk exhaustion, damaged files, process termination, offline behavior where claimed, and backup/recovery.

Execute native Windows and Apple Silicon protocols for whichever platforms are claimed. Cross-interface comparisons apply to workflows supported through GUI, CLI, and Python.

Keep measured performance separate from extrapolation. Review `ARCHITECTURE.md` and existing release evidence for outstanding platform and packaging limitations before approving claims.

19.10.1 Release acceptance checklist

[ ] Every in-scope requirement links to satisfactory evidence.
[ ] Every applicable reference case meets approved acceptance criteria.
[ ] No unexplained numerical discrepancies remain.
[ ] No unresolved defect can silently alter in-scope data, results, or quality decisions.
[ ] Every claimed environment has executed qualification evidence.
[ ] Required failure/recovery and record-handling tests pass.
[ ] Exclusions and deviations have documented rationale and approval.
[ ] Statistical and quality reviewers approve the conclusions.
[ ] Evidence identifies the exact released binaries and dependencies.

Do not approve on test-pass percentage or code coverage alone. Changes made after qualification require documented impact assessment and appropriate re-execution.

19.11 9. Handoff packages and ongoing maintenance

19.11.1 Supplier release-validation package

- Intended-use statement and validated feature/configuration matrix.
- Method specifications, references, limitations, and comparison policies.
- Requirements/risk/test traceability.
- Reference provenance, executed protocols, raw evidence, and result summaries.
- Deviation resolutions and known limitations.
- Release/dependency/environment identification.
- Approved validation summary report.

19.11.2 Customer qualification package

- Installation qualification: verify approved software and environment.
- Operational qualification: demonstrate required functions and controls.
- Performance qualification: demonstrate the customer's intended workflows.
- Templates for access, training, backup/restore, change control, and periodic review.
- Explicit supplier/customer responsibility matrix.

Customer qualification remains necessary for the customer's actual use. Supplier evidence reduces that work; it does not eliminate it.

19.11.3 Maintain the validated state

- Run affected tests for each change and the approved qualification suite for release candidates.
- Assess dependency, compiler, numerical-library, platform, and method-default changes.
- Version method profiles; do not silently reinterpret historical results.
- Preserve superseded evidence and reference baselines.
- Investigate discrepancies and assess impact on previously issued results.
- Define customer notification and correction procedures.
- Periodically review incidents, supported use, standards editions, and regulatory applicability.

19.12 Definition of done

A release is ready for a scoped validated-use claim when the approved requirements, methods, controls, workflows, and environments have satisfactory independently reviewed evidence, all relevant deviations are resolved or appropriately dispositioned, and the exact release plus customer responsibilities are documented.

Creating this plan or passing a development test suite does not itself satisfy that definition.

CHAPTER 20

Validation: requirements and risks

Engineering register, 2026-09-11. The user approved policy revision NIST-LRE-1; named owners and independent qualification approvals remain pending. Evidence IDs are emitted as `EVID-<case-id>-<case-hash-prefix>` within a timestamped execution report; retain the report and its baseline together.

Throughout the register, "Campaign report" refers to `evidence/statistical-suite-rehearsal.json`.

Requirement ID	Intended use / method	Risk IDs and rationale	Control / acceptance policy	Test IDs	Evidence / status	Accountable role / approver
REQ-STAT-DESC-001	Reference verification of sample mean/SD/-count	RISK-NUMERIC-001: cancellation or rounding can misstate variation	NIST-LRE-1 per-output minimum digits; exact counts	STAT-NIST-NUMACC1, STAT-NIST-NUMACC2, STAT-NIST-NUMACC3, STAT-NIST-NUMACC4	Engineering pass; DEV-NUM-001 closed as superseded policy finding; qualification pending	Statistical reviewer / unassigned
REQ-STAT-OLS-001	Reference verification of baseline one-predictor regression	RISK-NUMERIC-001: wrong estimates can mislead quality review	NIST-LRE-1 Norris coefficients/R-squared; exact count	STAT-NIST-NORRIS	Runner case evidence; engineering pass, qualification pending	Statistical reviewer / unassigned
REQ-STAT-ANOVA-001	Reference verification of baseline fixed-effects ANOVA	RISK-NUMERIC-001: incorrect F or degrees of freedom	NIST-LRE-1 SiRstv F; exact df/group/sample counts	STAT-NIST-SIRSTV	Runner case evidence; engineering pass, qualification pending	Statistical reviewer / unassigned
REQ-HARNESS-001	Trustworthy independent numerical evidence	RISK-REFERENCE-001: changed fixtures/mapping/settings or omitted outputs can falsely pass	Archived bytes, checksums, row/model checks, profile identity, required output sets, fail-closed evidence	All STAT-NIST cases; tests/validation/test_numerical.py fault cases	JSON/Markdown reports and focused test output; implemented	Validation engineer / independent quality reviewer unassigned
REQ-WORKFLOW-001	Correct included data from import through report/save/reopen	RISK-RECORD-001: mismatched population or stale results alter interpretation	Actual QML/controller/engine calibration protocol; exact rows, exclusion identities, snapshots and repeated reopen	WF-IMPORT-REPORT-001; tests/validation/test_industrial_workflow.py	Executable engineering proof; retain per-run results; independent qualification pending	Product and validation lead / unassigned

Requirement ID	Intended use / method	Risk IDs and rationale	Control / acceptance policy	Test IDs	Evidence / status	Accountable role / approver
REQ-DESKTOP-OLS-001	Desktop calibration estimates and ANOVA agree with independent references	RISK-NUMERIC-001: desktop implementation differs from baseline API	Retained NIST-LRE-1 thresholds plus WF-NORRIS-1 for newly covered outputs; three row orders	WF-IMPORT-REPORT-001; test_desktop_norris_precision_is_retained_after_reordering	See Chapter 24; independent review pending	Statistical reviewer / unassigned
REQ-REPORT-001	Preserve estimates, uncertainty and interpretations when sharing a fit	RISK-RECORD-001: omitted or partial report changes interpretation	Atomic multi-card addition; displayed parameter values checked; HTML interpretations and unchanged saved catalog	WF-IMPORT-REPORT-001; tests/unit/desktop/test_fit_report.py	Report omission fixed; permanent regression coverage	Product and validation lead / unassigned
REQ-CAPABILITY-001	Normal capability desktop workflow and model-check warning	RISK-NUMERIC-001 / RISK-RECORD-001: wrong index, model or population misleads process review	Fixed analytic references; explicit Normal model; report values and warnings; exclusion identity, save/reopen and rerun	WF-CAPABILITY-001; tests/validation/test_capability_workflow.py	Executable engineering protocol; retain per-run results; qualification pending	Statistical reviewer / unassigned
REQ-INTERPRET-001	Correct selected conclusions, uncertainty and unavailable-result explanations	RISK-NUMERIC-001: misleading explanation despite a computed statistic	Analytic OLS positive/negative/nonsignificant p/CI; ridge limitations; capability edge cases; Wilson interval	tests/validation/test_interpretation_correctness.py	Bounded method-level evidence, not product-wide interpretation qualification	Statistical reviewer / unassigned
REQ-STAT-NIST-001	NIST StRD univariate, regression, ANOVA and nonlinear datasets through the product	RISK-NUMERIC-001, RISK-REFERENCE-001	Certified values; NIST-LRE-1 minimum digits; exact counts	STAT-NIST-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-DESC-001	Descriptives, tables, distribution fit and diagnostics, probability, scoring and profiler	RISK-NUMERIC-001, RISK-REFERENCE-001	R references; 10^{-9} closed form, 10^{-6} iterative; exact counts	STAT-DESC-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-HYP-001	Hypothesis tests, power, bootstrap, permutation and comparisons	RISK-REFERENCE-001	R references; tail-probability comparisons; exact decisions	STAT-HYP-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-REG-001	Linear, generalized, mixed, panel, nonlinear, tree and selection models	RISK-REFERENCE-001, RISK-NUMERIC-001, RISK-BOUNDARY-001	R references; tiered tolerances; boundary and invalid inputs	STAT-REG-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned

Requirement ID	Intended use / method	Risk IDs and rationale	Control / acceptance policy	Test IDs	Evidence / status	Accountable role / approver
REQ-STAT-MV-001	Multivariate analysis and clustering	RISK-REFERENCE-001, RISK-NUMERIC-001, RISK-BOUNDARY-001, RISK-ROW-IDENTITY-001	R references; sign-invariant comparisons; row alignment	STAT-MV-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-SURV-001	Survival and reliability models	RISK-REFERENCE-001, RISK-NUMERIC-001	R `survival` references; 10^{-6} for fits	STAT-SURV-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-TS-001	Correlations, forecasts, ARIMA, dynamic regression, VAR/VECM, GARCH, Zivot–Andrews and rolling validation	RISK-NUMERIC-001, RISK-REFERENCE-001	R `stats`, forecast, vars, urca, rugarch; analytic benchmarks	STAT-TS-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-QUAL-001	Specifications, control charts and capability	RISK-NUMERIC-001, RISK-REFERENCE-001	R and analytic SPC constants and rules; exact flagged positions	STAT-QUAL-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-MSA-001	Gage R&R, gage studies, attribute agreement, Deming and manufacturing decisions	RISK-NUMERIC-001	R `aov`, `lme4`, `irr`; analytic rules	STAT-MSA-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-DOE-001	DOE, power and sample size, reliability plans and worksheets	RISK-NUMERIC-001, RISK-REFERENCE-001	R `lm`, `pwr`, analytic plans; exact integers	STAT-DOE-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-EXP-001	Explore plots, variation charts, Pareto, multimodality and graphs	RISK-NUMERIC-001, RISK-REFERENCE-001	R and analytic references; conventions noted	STAT-EXP-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-PREP-001	Prepare operations, reshape and joins	RISK-NUMERIC-001	Analytic tables compared exactly, with source-row lineage	STAT-PREP-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-HARVEST-001	Published values moved out of unit tests	RISK-REFERENCE-001	Published values at their stated precision	STAT-HARV-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned

Requirement ID	Intended use / method	Risk IDs and rationale	Control / acceptance policy	Test IDs	Evidence / status	Accountable role / approver
REQ-STAT-SIM-001	Coverage, Type I error and achieved power without closed forms	RISK-NUMERIC-001	Pre-registered specifications; nominal $\pm$ 3 binomial SEs; no reruns	STAT-SIM-*	Campaign report; engineering evidence; qualification pending	Statistical reviewer / unassigned
REQ-STAT-WIRE-001	Every statistical menu item reaches its validated entry point with the chosen settings	RISK-RECORD-001	Real QML window and engine; recorded settings and displayed values compared	tests/validation/wiring; `inventory/wiring.json`	Campaign report; engineering evidence; qualification pending	Product and validation lead / unassigned
REQ-EVIDENCE-001	Retained, baseline-identified source and installed-wheel workflow evidence	RISK-REFERENCE-001: lost failures or source-path substitution yields misleading acceptance	Non-overwrite directories; subprocess failure/timeout retention; manifest/hash checks; isolated wheel imports and immutable source baseline	tools/run_validation_evidence.py; tools/run_release_validation.py; archive/release fault tests	CI archival configured; controlled release custody and native acceptance remain open	Release engineer / independent quality reviewer unassigned

20.1 Initial record-control gap assessment

This is a source-based engineering assessment against the plan, not a regulatory applicability conclusion. See the root `ARCHITECTURE.md` for existing controls and release limitations. Customer responsibilities below are proposed and unverified.

Control area	Existing foundation	Remaining gap / proposed responsibility
Data/result/settings identity	Profiles, population receipts, checksummed artifacts and saved semantic reports	Coverage across desktop methods and complete workflows must be demonstrated by HypoNum
Original data and exclusions	Source references/checksums, optional embedding, row-exclusion persistence	Qualify repeated saves, sorting, same-count exclusion swaps, and original-data retention; HypoNum plus customer retention procedures
Backup/restoration	Atomic save, autosave/recovery and existing reliability runner	Execute actual customer backup/restore and retention arrangements; shared responsibility
Attributable/protected audit history	Serializable workflow/provenance gives useful history	Does not establish protected who/when/why audit trails; quality lead must select controls or restrict intended use
Access, approvals, electronic signatures	Optional Enterprise contracts are service-neutral mocks/signing primitives	No evidence of a complete regulated-record authorization/signature workflow; applicability and controlled-system integration remain open

Control area	Existing foundation	Remaining gap / proposed responsibility
Plugins / unapproved methods	Explicit local discovery and spawned execution; plugins are trusted code, not sandboxed	Customer installation policy and enforced scope restrictions need definition and verification
Released environment	Existing packaging/release validation utilities	Signed Windows and notarized native Apple Silicon clean-machine evidence remain outstanding in the architecture

RISK-RECORD-001 and RISK-REFERENCE-001 remain open until controls, accountable owners, and retained evidence are independently reviewed. Risk severity/likelihood and regulatory applicability require the actual intended use and qualified human review.

CHAPTER 21

Validation: numerical policy and record

Status: engineering implementation, 2026-09-11. The user approved the NIST-LRE-1 comparison-policy revision in this session. Release scope and independent statistical/quality qualification approval remain pending. This starts VAL-02/VAL-03 of the Chapter 19. It does not complete the first workflow milestone, freeze a release, or qualify any customer deployment.

21.1 Execute and retain evidence

From the repository, using the locked environment:

```
uv sync --locked --all-groups
uv run --locked python tools/run_validation.py --output-dir .tmp/numerical-evidence
uv run --locked python tools/run_validation.py --qualification --output-dir .tmp/
    numerical-prerequisites
uv run --locked pytest tests/validation tests/unit/analysis/test_goldens.py tests/unit
    /analysis/test_profiles.py tests/unit/analysis/test_methods.py tests/unit/analysis/
    test_baseline_methods.py
```

Use a new output directory for every run. `results.json` retains comparisons, original decimal expectations, absolute/relative error, LRE scores, acceptance limits, case and artifact hashes, traceability, full existing analysis goldens/receipts, timing, source commit, working-tree file hashes, lock hash, installed dependency versions, OS, architecture, and Python version. `summary.md` is the readable companion. The scoped algorithms use Python statistics/math; they do not use BLAS. The evidence records installed versions separately from the lock identity; the runner does not claim it has established a clean or reproducible environment. Commit the reviewed baseline and retain its environment/build before qualification.

Exit 0 means all required engineering cases passed. Exit 1 means failed or blocked evidence, including missing files, malformed/empty suites, missing required outputs, or pending independent review in `--qualification` mode. There are no silent skips. Unsupported reference outputs are listed explicitly in every case and are outside these API comparisons. `--qualification` is only a prerequisite check, not a release approval workflow. All six engineering cases pass under NIST-LRE-1.

Store executed evidence with the matching source checkout in controlled release storage; `.tmp` is an ignored local work area, not the final evidence repository. The original validation plan was already untracked and is intentionally unchanged.

21.2 Desktop Help document

Help > Validation opens a read-only, modeless document with an overview of all six reference tests, the 21 individual comparisons, accuracy scores and minimums, error measurements, execution date, and tested source/environment. It opens offline without importing data or running tests on the UI thread. The bundled document is an identified execution record, not a live assertion about the current project.

The initial document uses the passing execution for source `c277457`. Refresh it from a retained report when new validation evidence is published:

```
uv run --locked python tools/build_validation_document.py --result .tmp/numerical-
    evidence/results.json --output qml/documents/validation.html
```

Review the generated document together with that report. The generator includes every case, including failures/blocked cases if present, and rejects inconsistent passed statuses. It does not synthesize test results. The document ships in the existing QML resource directory in wheels and Windows/macOS bundles, so an installed application does not depend on a repository checkout, `.tmp`, or network access.

21.3 Candidate scope and method specification

Intended engineering use: compare baseline public Python analysis APIs with independent NIST references before selecting a customer quality-decision workflow. No industry, jurisdiction, approved operating environment, or customer use has yet been selected. Responsible roles below are proposed; named owners remain unassigned.

API/configuration	Reference cases	Compared outputs	Scope status
`analysis.methods.descriptives`, sample variance denominator n-1, no quantiles	NumAcc1-4	mean, sample standard deviation, count	Engineering pass under NIST-LRE-1; qualification pending
`analysis.baseline_methods.ordinary_least_squares`, one predictor and intercept	Norris	intercept, slope, R-squared, count	In engineering validation; review pending
`analysis.baseline_methods.one_way_anova`, classical fixed effects	SiRstv	F, between/within df, group count, count	In engineering validation; review pending
Desktop Fit Y by X (`compute_fits`) and hypothesis ANOVA	Not yet adapted	Full desktop outputs including standard errors/ANOVA tables	Excluded from this increment; separate implementations
Import, prepare/exclude, reports, save/reopen, GUI actions, plugins, other methods, release installers	Not yet executed here	Workflow and release evidence	Excluded from this increment; remains planned

All cases bind `tabelz-transparent` version `1.0.0` and its complete settings hash. Inputs are finite binary64 conversions of the original decimal strings, with no missing values or exclusions. The runner checks the exact original column order: univariate Y, Norris Y/X, and SiRstv group/Y. It requires an OK result and no exclusions. The archived counts and dimensions bound this initial domain; passing these cases does not establish general input-size or numerical-domain limits.

21.4 Comparison policy NIST-LRE-1

Continuous NIST outputs use logarithmic relative error (LRE), capped at 15 digits, following the reporting convention in the Stata 18 NIST evaluation. For expected value `c` and actual value `a`:

```
absolute_error = abs(a - c)
relative_error = absolute_error / abs(c)                when c != 0
LRE = min(15, -log10(relative_error))                   when c != 0
LRE = min(15, -log10(absolute_error))                   when c == 0
LRE = 15                                               when a == c
```

The comparator converts binary64 results to their exact decimal values and retains reference decimal strings, using a 60-digit decimal context for errors/logarithms. Absolute and relative errors remain evidence. For each output the case declares `minimum_digits`; acceptance compares the unrounded error to `abs(c)*10**(-minimum_digits)` (or `10**(-minimum_digits)` for a zero reference). This avoids a rounded displayed LRE turning a near-boundary failure into a pass. LRE has no lower clamp; grossly inaccurate finite values can score below zero. Missing/non-finite values fail and receive no LRE score. Exact integer count comparisons remain exact, with no LRE acceptance rule.

The minimums below retain the whole-digit part of each published Stata result, capped at 15. This is an explicit HypoNum engineering acceptance rule inspired by external benchmarks; the vendors publish scores, not these pass thresholds. Taking the whole-digit part avoids treating their rounded scores as exact bounds. These minimums are fixed in fixtures, never inferred from the HypoNum run.

Cases / output	Published Stata 18 LRE	Required minimum digits
NumAcc1-4 mean	15.0	15
NumAcc1-2 sample SD	15.0	15
NumAcc3 sample SD	9.5	9
NumAcc4 sample SD	8.3	8
Norris intercept	12.8	12
Norris slope	14.4	14
Norris R-squared	15.5 in detailed log, capped at 15	15
SiRstv F	13.1	13

Sources: Stata's univariate and ANOVA tables linked above, and its Norris execution log. The Stata benchmark identifies its execution as Stata/SE 18 on Linux, 2023-04-06. The reporting convention is also used in the JMP 16 accuracy report. Comparative scores do not establish universal suitability: a large mean can have high relative accuracy while a small spread is less accurate. We therefore retain separate SD criteria and all raw errors. Decision boundaries, tail probabilities, undefined results, and customer-use-specific criteria remain separate work.

Existing `analysis.goldens` verifies result/receipt structure and fingerprints; the new decimal comparator establishes agreement with independent reference values. No expected values are calculated by the implementation under test.

21.5 Policy revision and disposition

On 2026-09-11 the user explicitly approved replacing the arbitrary absolute cutoff with the proposed NIST accuracy scoring policy. No analytical implementation, input, or certified expected value changed. Case versions increased from 1.0.0 to 1.1.0; expectation and case hashes were updated. Evidence report schema 1.1.0 adds the LRE score and selected minimum while retaining error measurements.

The superseded suite, case records, and expected policies are preserved in `tests/validation/reference_data/history/ABSREL-1.json`. The original raw and normalized inputs retain their hashes. Earlier local executions in `.tmp/validation-engineering-final` and `.tmp/validation-prerequisites-final` remain untouched; their outcomes reflect the superseded policy. Legacy continuous comparisons remain supported for interpreting/replaying that policy. New executions use new output directories; earlier failed reports are never rewritten as passes.

ID	Case/output	Observation	Disposition
DEV-NUM-001	NumAcc4 / mean	NIST decimal 10000000.2; Python 3.14.3 result 10000000.200000001; absolute error 1.1175870895385742187 5e-9; capped LRE 15	Closed as a superseded engineering-policy finding. The arbitrary 1e-9 cutoff is withdrawn; this is not an established numerical defect. Passes NIST-LRE-1.

All six cases now execute as ordinary passing tests; there is no NumAcc4 xfail or special case in the comparator. Fault injection and LRE boundary tests still fail altered or insufficiently accurate results. The engineering runner exits 0; qualification prerequisite checks remain blocked solely on independent review records. No other platform was run, and customer-release qualification is pending.

21.6 Next increment

The separate Chapter 24 now exercises the actual desktop OLS calibration path, preparation/exclusions, report export and repeated save/reopen. It adds independent checks for coefficient standard errors and ANOVA outputs. The baseline suite and its historical scope above are unchanged; the new protocol has its own runner and retained evidence. Hypothesis-ANOVA desktop qualification remains outside that increment.

1. Assign quality/product/statistical owners; approve intended use and release scope.
2. Independently review case references and NIST-LRE-1 minimums for qualification.
3. Adapt actual desktop statistical engines, then expand applicable NIST cases.
4. Execute one actual import-to-report/save-reopen protocol with exclusions, linked permanent defect regressions, and GUI evidence.
5. Complete record-control applicability and clean-machine release qualification.

CHAPTER 22

Statistical reference validation

Audit provenance. The September 2026 reference audit covered issues 39 and 56–76. Codex (GPT-6) wrote the chapter and performed the statistical review; GPT-6 Sol implemented the corrections and retained fixtures; Claude Fable 5.1 and the project owner supplied source discovery and initial checks. This chapter preserves the internal evidence from the former standalone audit.

22.1 Purpose and scope

Statistical validation begins by identifying the calculation being claimed. Two implementations can use the same familiar method name and still differ in their estimator, probability model, starting conditions, finite-sample correction or treatment of an undefined result. Agreement is meaningful only after those choices have been aligned.

This chapter records the reference audit of selected control charts, measurement-system calculations, process capability, distribution diagnostics, response optimization and multiple comparisons. The audit has two practical outputs: corrections to HypoNum where the stated calculation was not performed, and retained evidence that links each assessed calculation to an identifiable source and a reproducible comparison. The initiating discussions are GitHub issues 39 and 56–76, indexed by tracking issue 77. The issue history records decisions; this chapter and the versioned fixtures record their technical basis and executed scope.

The treatment is self-contained. Necessary definitions, assumptions and worked comparisons are given here rather than requiring the reader to reconstruct them from a user manual. The manual remains the authority for the product's supported behavior. This chapter explains how selected claims about that behavior were assessed. It does not extend the supported method catalogue or certify every setting, dataset, platform or intended use.

22.1.1 Four distinct conclusions

Corroborated definition.
The formula and its assumptions have been checked against an identified publication or authoritative methods description. This establishes the method being implemented, not the accuracy of a software execution.

Checked implementation.
The relevant production calculation and boundary behavior have been compared with a justified numerical example or invariant. An identified discrepancy is corrected and retained as a regression case.

Executed independent comparison.
A case has been run through the production entry point and compared with published values, a separately implemented reference or an independently verified derivation. The case, settings, expected and actual values, acceptance rule and execution baseline are retained.

Release qualification.
This requires the wider approved scope, environment and release evidence. The selected reference comparisons in this chapter do not establish that conclusion by themselves.

22.1.2 How to read a comparison

A published table is retained at its original precision. A value recomputed from the table's inputs is a separate result, even when it uses the paper's formula. A value produced by an independent R package is also identified as such. These distinctions prevent a formula copied from the product manual from being presented as an independent published numerical standard.

Every numerical comparison must specify the population, data order, units, sample-size convention, fitted or supplied parameters, probability tail, confidence level and supported output. A reference for a point estimate does not automatically validate its interval, optimizer, missing-value handling or GUI. The method sections below identify these boundaries explicitly.

22.2 Evidence register and reproducibility

The execution register identifies the assessed source, reference versions, case selection and comparison outcomes. The assessment concerns a reviewed working copy awaiting integration. Successful selected checks establish the recorded comparisons; they do not establish release qualification.

22.2.1 Reference classes

The repository's existing statistical-validation harness distinguishes *certified*, *published*, *independent implementation* and *analytic* references. A NIST explanatory webpage is not necessarily a certified numerical dataset. A published equation can justify an analytic calculation without making its computed digits published observations.

Independence is assessed for the compared output. For example, an external Deming implementation can supply slope and interval expectations while row identity and minimum-sample refusals remain HypoNum contract checks. A probit fit and the Fieller transformation of its coefficients can have different reference sources within the same study. A package name at the case level must not conceal that distinction.

22.2.2 Precision and discrepancy treatment

The original inputs and printed expectations are preserved. Where source inputs are rounded, a discrepancy can arise before the implementation is called. The investigation distinguishes an implementation defect, a method convention, reference precision below the comparison tier, and a suspected source error. A failed comparison is retained with its disposition; its tolerance is not enlarged merely to obtain a pass.

Full-precision independent expectations use the declared numerical tier. Printed-value comparisons use a separately justified precision treatment. Counts, method labels, unavailable states and signal identities are exact contract outputs. An unavailable statistic is not replaced by an invented finite value, and an execution failure is not silently omitted from totals.

22.2.3 Reproduction path

Reference generators and pinned R dependencies belong under `tests/validation/generators/r/`. The indexed datasets, cases and expected outputs belong under `tests/validation/reference_data/`. Cases execute through `tools/run_statistical_validation.py`, using the same

engine bindings as the application. The executed evidence must name the exact case selection; a selected run is not a full-suite result.

The source papers are identified in Section 22.6. The audit uses the project owner's supplied copies and authoritative online documentation. Whole copyrighted papers are not required in a software distribution. Source identities and hashes, lawful reference inputs, transformations and numerical expectations provide the reproducibility link.

22.2.4 Method and source register

Table 22.1 maps every initiating issue to the precise claim assessed. Numerical execution IDs and outcomes are recorded separately below; this table identifies the source relationship and its limits. In particular, a source-backed formula and a published dataset are not interchangeable reference classes.

Table 22.1. Reference scope by method. Bracketed numbers identify entries in Section 22.6.

Issue	Method	Source location	Compared claim and evidence boundary
56	Shewhart $\overline{X}$–R, $\overline{X}$–S, I–MR	NIST 6.3.2.1–2 [1]	Normal-range constants, scale estimators and limits. Printed constants and full-precision derived expectations remain distinct.
57	Nelson rules	Nelson Fig. 1, p. 238 [2]	Rule 8 needs both sides in the eight-point window. Constructed decision cases and saved-analysis compatibility check the correction.
58	p, c, np, u charts	NIST 6.3.3.1–2 [3]	Published wafer counts check c limits/signals; p equations corroborated. np, u and varying-denominator cases are derived extensions.
59	Tabular CUSUM	NIST 6.3.2.3 [4]	Recursion, zero start, K/H units and strict signal boundary. Known-parameter run-length claims do not transfer automatically to estimated baselines.
60	EWMA	Lucas–Saccucci Table 1 [5]; Montgomery Example 9.2 [6]	Published sequences check recursion and applicable limits. Finite-time and asymptotic startup conventions are recorded separately.
61	Normal tolerance factors	Howe (2.13), Table 1 [12]; `tolerance` [13]	HE2 retained with honest approximation label; one-sided noncentral-t factor distinct. Source coverage values are not a new simulation.
62	Grubbs test	NIST 1.3.5.17.1 [21]	Single-candidate two-sided statistic, critical value and inverse probability. No iterative multiple-outlier claim.
63	Normal Anderson–Darling	Stephens illustration and table [22]; `nortest` [23]	Raw statistic and later calibration checked separately. The 1974 and later 1% decisions differ for the printed example.
64	Desirability	Derringer Figs. 1–2 [14]; NIST 5.5.3.2.2 [15]	Linear individual functions, importance-weighted product and exact zero propagation. Optimizer ranking and zero-result reporting checked.

Reference scope by method, continued

Issue	Method	Source location	Compared claim and evidence boundary
65	Capability indices and SE	NIST 6.1.6 [31]; Bissell Tables 3–4 [16]	Index/estimator pairing, effective df and approximate uncertainty. Printed 2 SE intervals are distinct from requested normal quantiles.
66	Short-run Z–MR	Minitab methods [7]	Known part centres/scales, standardized limits and span-two MR constants. No qualification of alternative parameter estimators.
67	Rare-event g, t charts	Benneyan, p. 311 [8]	Support translation and mean/MLE/probability-limit convention explicit. Benneyan's three-sigma example is retained as a different construction.
68	Johnson S_U, S_B, S_L	Slifker–Shapiro (1) and examples [24]; Minitab [32]	Parameter transformations checked at printed precision. Supplied-parameter examples do not validate maximum-likelihood fitting or automatic selection.
69	Binomial OC curve	NIST $n = 52, c = 3$ example [17]	Inclusive acceptance probability. A printed discrepancy at $p = 0.05$ is retained alongside a direct binomial sum.
70	Type 1 gage	Minitab equations and bias test [18]	Inspected C_g/C_{gk} images support the formulas; constructed numerical case checks indices, tolerance percentage and bias inference.
71	Deming regression	NCSS pp. 303-1–303-6 [19]; `mcr` [20]	Reciprocal variance-ratio mapping, coefficients and jackknife SE/intervals. Application-specific refusal/row contracts are separate.
72	Laney p', u'	Laney pp. 533–535 [9]	Standardized moving-range dispersion, adjacency and unavailable-state correction. Positive underdispersion retained.
73	MEWMA	Lowry et al. (2.3)–(2.5), Table 4 [10]	Constructed exact baseline plus published monitoring inputs. Printed T^2 and rounded-input recomputation are separate fields.
74	Subgroup T^2 and determinant chart	Montgomery (11.21), Example 11.1 [6]; Alt–Grimshaw pp. 1016–1017 [11]	Phase II critical value and b_1/b_2 moments. Reconstructed raw groups are not presented as original observations.
75	Moving average	Montgomery (9.37)–(9.40), Example 9.3 [6]	Thirty-point known-parameter example, expanding startup and signal identity. Withheld startup limits are a separate option.
76	Fieller inverse calibration	Franz pp. 7–8 [25]; `twopartm` [26]	Bounded ratio endpoints and covariance sign; unavailable bounded interval reporting. Full-set shape is not a displayed product output.

Reference scope by method, continued

Issue	Method	Source location	Compared claim and evidence boundary
39	Future GT2/Gabriel contract	Hochberg (3.2), (4.4) [27]; Gabriel pp. 724–725 [28]	SMM index, critical-value integration and printed example; false ratio invariant removed. Contract evidence, not implemented F137 qualification.

22.2.5 Executed independent comparisons

The retained file `docs/engineering/validation/evidence/reference-inference-oracles.json` contains per-output results from the production calculations and the named independent references. Table 22.2 gives representative comparisons; errors are calculated from the retained full-precision numbers, not from the shortened display values in this table.

Table 22.2. Executed comparisons against pinned independent packages. Each row uses an absolute acceptance tolerance of 10^{-8}.

Output	HypoNum	Independent reference	Absolute error
HE2 factor	3.08208866133824	3.08208866133824	1.33×10^{-15}
AD raw A^2	0.946771879598888	0.946771879598886	1.78×10^{-15}
AD later p	0.010454024005148	0.010454024005148	1.20×10^{-16}
Fieller lower	0.644085519308482	0.644085519308482	2.22×10^{-16}
Fieller upper	3.493324927434632	3.493324927434631	4.44×10^{-16}

The IDs are `REF-HOWE-HE2-001`, `REF-AD-STEPHENS-001` and `REF-FIELLER-BOUND-001`. The package versions are `tolerance` 3.0.0, `nortest` 1.0-4 and `twopartm` 0.1.0. The same manifest retains `REF-OC-NIST-001`: the binomial sums at $p = 0.01, 0.03$ and 0.05 agree with an independent direct sum within 10^{-14}. That is a passed derived comparison; it does not turn the disagreeing printed 0.739 into a passed published-value comparison.

The Gabriel contract calculation is retained in `docs/engineering/validation/evidence/f137-gabriel-contract.json`. It records the SMM integration, contrast endpoints, balanced-identity convention and the numerical counterexample to the false GT2/Tukey ratio bound. There is no production F137 result to compare: its status is explicitly contract-only.

22.2.6 Executed selection and acceptance

The final indexed selections contain 149 distinct cases, all passed, with zero failed, blocked or unsupported cases. Table 22.3 gives their exact scope. The family selections include neighboring existing fixtures; 149 is a count of executed input/settings cases, not a count of independent publications or newly qualified methods. The supplementary direct comparisons, focused regression tests and future Gabriel contract calculation are not added to that indexed-case total.

The corresponding `results.json` files are retained under `.tmp/artifacts/` in `reference-quality-post-extraction-20260927/`, `reference-measurement-accepted-20260927/`, `reference-grubbs-source-run-20260927/`, `reference-stephens-ad-run-20260927/` and `reference-doe-tolerance-run-20260927/`, respectively. Each compared output retains its expected value, actual value, comparison rule and result. Numeric tolerances apply per output; successful error/refusal cases instead require the declared contract response.

After the SPC rule extraction, the selected numerical unit checks passed 103 tests. The selected recipe, controller and saved-study replay checks passed 19 tests. The extraction preserves the

Table 22.3. Final indexed numerical-reference selections. Each row has zero failed, blocked and unsupported cases.

Selection	Passed cases
Quality family	97
Measurement family	49
NIST Grubbs example	1
Stephens normal AD example	1
DOE HE2 tolerance planning	1
Total distinct cases	149

old helper import identities through explicit re-exports. The subsequent wording-only docstring edit was followed by the final quality reference run and the recipe/controller/replay checks. Formatting and lint checks passed, as did mypy on 995 source files and the architecture check on 1,345 source files. The manifest identifies the separate DOE, profiler, diagnostic and Fieller acceptance checks and their recorded scope; these selected test counts are not summed into a full-suite claim.

For a new reproduction from the repository root, use the locked environment and a new output directory. For example:

```
uv run --locked python tools/run_statistical_validation.py run --family quality `
  --output-dir .tmp/artifacts/reference-quality-recheck
uv run --locked python tools/run_statistical_validation.py run --family measurement `
  --output-dir .tmp/artifacts/reference-measurement-recheck
```

The other three selections use `--case` with `STAT-HARV-NIST-GRUBBS-001`, `STAT-DESC-AD-STEPHENS-001` and `STAT-DOE-PREC-TOL-001`. The retained command manifest gives the recovered assessment invocations, including focused tests, and separately labelled reproduction commands for reference generators. Where an earlier invocation or log was not retained, the manifest says so rather than presenting a reconstructed command as an original execution record. Re-running a case is distinct from regenerating its expectation; the latter requires review of the source, generator and resulting evidence.

22.2.7 Execution baseline and retained records

The assessment was performed in the dedicated `feature/reference-validation-20260927` worktree, based on commit `9343207a6ba661759fa51dde1667dd58966ac114`. The corrected source was uncommitted at the assessment checkpoint. That base commit alone therefore does not identify the tested implementation. The retained `reference-change-manifest.json` records SHA256 hashes of the changed implementation, generators, tests and evidence, together with the exact execution commands and result files. Integration or a later source change requires its own acceptance assessment.

Three records under `docs/engineering/validation/evidence/` provide different parts of the audit trail:

`reference-source-inventory.json`
Identities, file hashes and page counts of the owner's supplied papers. The incomplete and complete Lucas–Saccucci copies are distinguished.

`reference-campaign-matrix.json`
The issue-to-method mapping, production entry points, case IDs, source classes, execution results, compared outputs, errors, acceptance thresholds and method-specific limitations.

`reference-change-manifest.json`
The source and dependency checkpoint, exact commands, test counts and hashes of retained execution reports. This identifies the uncommitted assessment without presenting it as a released build.

The numerical environment is Windows 11, Python 3.13.15, SciPy 1.18.1 and R 4.6.1. The four named R references are pinned to `mcr` 1.3.3.1, `tolerance` 3.0.0, `nortest` 1.0-4 and `twopartm` 0.1.0. The `uv.lock` SHA256 is `d061c6d8307be653e7133fd2a98982a7806e78845410358d22fba1364c3f6def`; the R lockfile SHA256 is `805355ade32c4d081ade2451653bbf57a6890c675d2ed67234e4519732ed3f63`. Package versions identify an oracle implementation; they do not, by themselves, establish its suitability for every boundary case.

22.2.8 Retained case index

Table 22.4 identifies the principal evidence for each issue. The campaign matrix also lists the pre-existing cases whose source provenance was refreshed. A case ID identifies a particular calculation, not every setting offered by that method. Focused regression tests are identified separately from the numerical-reference cases.

Table 22.4. Principal retained evidence by initiating issue.

Issue	Calculation	Case or focused regression evidence
56	Shewhart constants	`STAT-QUAL-PUBCHART-001`.
57	Nelson rule 8	`test_spc_zone_rules.py`; corrected and historical operation-version checks in `test_recipe_quality.py`.
58	Published *c* chart	`STAT-QUAL-PUBCHART-002`.
59	CUSUM	`STAT-QUAL-PUBCHART-003`.
60	Two EWMA sequences	`STAT-QUAL-PUBCHART-004` (Montgomery) and `STAT-QUAL-PUBCHART-006` (Lucas–Saccucci).
61	HE2 and one-sided factors	`STAT-MSA-TOL-004`, `STAT-DOE-PREC-TOL-001`, `REF-HOWE-HE2-001`; one-sided check `STAT-MSA-TOL-001`.
62	Two-sided Grubbs	`STAT-HARV-NIST-GRUBBS-001`.
63	Normal AD	`STAT-DESC-AD-STEPHENS-001` and `REF-AD-STEPHENS-001`.
64	Desirability and ranking	`test_doe_advanced.py`, `test_model_optimization.py`, `test_stored_model_profiler.py`; recorded-fit QML workflow and versioned recipe checks.
65	Four capability estimators	`STAT-QUAL-BISSELL-001` through `STAT-QUAL-BISSELL-004`.
66	Short-run Z–MR	`STAT-QUAL-PUBCHART-007`.
67	Translated *g* chart	`STAT-QUAL-PUBCHART-008`. The *t*-chart convention has focused functional coverage, rather than a new independent published waiting-time fixture.
68	Supplied Johnson parameters	`STAT-QUAL-JOHNSON-PARAM-001` and `STAT-QUAL-JOHNSON-PARAM-002`; transformation only.
69	Binomial OC	`STAT-MSA-OC-NIST-001` and `REF-OC-NIST-001`.
70	Type 1 gage	`STAT-MSA-TYPE1-001`.

Principal retained evidence, continued

Issue	Calculation	Case or focused regression evidence
71	Deming and jackknife	`STAT-MSA-DEMING-001` through `STAT-MSA-DEMING-009`; the matrix distinguishes successful `mcr` comparisons from invalid-input and row-policy checks.
72	Laney dispersion	Existing `STAT-QUAL-ATTR-002` and `STAT-QUAL-ATTR-005`; focused zero/adjacency, controller and historical-replay regressions.
73	MEWMA sequence	`STAT-QUAL-PUBCHART-009`; all ten public T^2 values and signals, with reference Z vectors retained separately.
74	Subgroup Phase II limit	`STAT-QUAL-PUBCHART-010`; existing subgroup/determinant cases retain their separate output scope.
75	Moving-average sequence	`STAT-QUAL-PUBCHART-005`; all thirty averages, startup limits and absence of breaches.
76	Bounded Fieller intervals	`STAT-MSA-RESP-001` through `STAT-MSA-RESP-004`, plus `REF-FIELLER-BOUND-001` and explicit unavailable-boundary tests.
39	Future Gabriel/GT2 contract	`REF-F137-GABRIEL` in `f137-gabriel-contract.json`; independent quadrature and the printed example, with no implemented F137 result.

22.2.9 Limits of the assessment and earlier failed attempts

The retained examples qualify their declared calculations. They do not establish general confidence-interval coverage, false-alarm rates with estimated parameters, Johnson maximum-likelihood fitting, every chart option, or every supported dataset. The rare-event t chart has a documented convention and focused functional checks, but no new independently computed published waiting-time example. F137 remains a future implementation. Full application regression, wider coverage simulations and release qualification were outside this selected assessment.

One older stored-profiler integration test, `test_stored_model_profiler_workflow_qml.py::test_menu_controls_mean_profiles_report_reopen_replay_and_invalidation`, fails when it restores a directly constructed artifact without the named fitted-result dependency now required by the workflow. The same failure was reproduced on the unchanged source baseline with the locked environment. It remains a known baseline test failure. The new supported-path test, `test_recorded_fit_all_zero_goal_shows_no_acceptable_candidate`, uses an actual recorded fit and verifies the menu, setup, calculation and visible all-zero result. Its pass is reported separately; it does not convert the older failure into a full GUI-suite pass.

Earlier reference runs also encountered stale generator/case hashes while fixture metadata was still changing, and one Bissell case expected a padded df string rather than the actual public description. Those unsuccessful runs are retained separately. Hashes and the fixture formatting were corrected before the final accepted selections. Expected numerical values and tolerances were not weakened to make those failures disappear.

22.2.10 Reference precision requiring an explicit disposition

Table 22.5 records differences that would be obscured by assigning one broad tolerance to both a printed table and a separately derived expectation. The recomputed values below are rounded for display; the retained machine-readable evidence preserves their numerical precision.

Table 22.5. Selected source and convention differences. A discrepancy is retained with its explanation, rather than recast as a passed printed-value comparison.

Quantity	Published and recomputed values	Disposition
Shewhart $D_4(5)$	NIST 2.115; normal-range calculation 2.114499145	Retain the primary definition and the full-precision constants. The printed value is not ordinary three-decimal rounding of the latter.
Binomial OC, $n = 52, c = 3, p = 0.05$	NIST 0.739; direct sum 0.738317411	Printed comparison disagrees. The independent full-precision sum is the reference expectation; finite-lot behavior is outside this example.
Lowry MEWMA, rows 2 and 9	Printed 3.17 and 9.65; recomputed 3.177196 and 9.644983	Use the supplied rounded monitoring coordinates and retain both sets of values. An enlarged two-decimal tolerance would hide the source precision issue.
Subgroup Hotelling limit	Montgomery 15.16; quantile calculation 15.165030183	The published example uses approximate intermediate arithmetic. The full-precision limit rounds to 15.17.
Bissell pooled-scale SE	Table 4 prints 0.086; recomputation approximately 0.086805064	Keep the source entry and derived result separate. The source's ± 2 SE interval also differs from a 95% normal-quantile interval.
Grubbs critical value	NIST example 2.032; retained two-sided case 2.126645087	Different hypotheses: the source worked example is upper one-sided; the retained case applies the general two-sided formula to the same data.
Anderson–Darling at 1%	Stephens 1974 rejects; later `nortest` calibration gives $p = 0.010454024$	Different finite-sample calibrations. Retain the explicitly named later convention and its non-rejection at 1%.
Howe achieved confidence	Table 1: λ_1 gives 0.9486; λ_2 gives 0.9485	The earlier issue summary attributed the second value to the first. The source column for the retained HE2 form is λ_1; neither value supports a uniform conservative guarantee.
Gabriel fourth lower endpoint	Printed 1.429; full-precision recomputation 1.429631505	Printed inputs and critical-value rounding are retained separately. This is evidence for the future method contract, not an implemented comparison procedure.

22.2.11 Corrected analyses and saved-study compatibility

New `quality.spc` requests use implementation version 1.1.0. They apply the corrected Nelson rule 8 and refuse an unavailable Laney dispersion estimate with an explanatory message. Laney moving-range pairs also respect gaps left by excluded or omitted source rows. The GUI and recorded replay path preserve those positions before filtering.

The registered 1.0.0 adapter retains the historical rule 8 predicate and Laney fallback for replay of an already recorded computation. Preserving that history does not make its statistical defect correct. The corrected calculation requires a new analysis; a replay is expected to reproduce the recorded implementation. Shared historical rule labels avoid asserting the new both-sides condition for an old result.

New `doe.optimize` requests likewise use implementation version 1.1.0. They propagate a zero desirability exactly and identify an all-zero search as `no_acceptable_candidate`. Strictly positive representable values, including subnormal values, remain eligible; no arbitrary small-positive cutoff replaces the removed zero floor. The 1.0.0 adapter reproduces the earlier floor for saved requests. The profiler paths communicate an all-zero outcome through their existing warning fields.

Only these explicitly designated current adapters change selection for a new request. Other analysis families retain their existing unique-adapter requirement. Implementation-version strings remain identifiers; a general numeric ordering of those strings is not introduced by this correction.

22.3 Statistical process-control charts

This section distinguishes a chart's plotted statistic, parameter estimator, control limit and signalling rule. Matching one of those elements does not establish agreement in the others. In particular, a known-parameter chart and a chart fitted to a short Phase-I baseline do not automatically share the same false-alarm or average-run-length properties.

22.3.1 Shewhart limits and normal range constants (issue 56)

For independent normal observations in subgroups of size n, let $d_2(n)$ and $d_3(n)$ be the mean and standard deviation of the range of n independent standard normal variables. Let $c_4(n) = E(s)/\sigma$, where s uses divisor $n - 1$. The range and standard-deviation estimators are

$$\hat{\sigma}_R = \frac{\overline{R}}{d_2}, \qquad \hat{\sigma}_S = \frac{\overline{s}}{c_4}.$$

The corresponding limits for subgroup means are $\overline{\overline{x}} \pm 3\hat{\sigma}/\sqrt{n}$. Thus the familiar $A_2 = 3/(d_2\sqrt{n})$ is a derived constant, not a separate probability model. The dispersion-chart limits are

$$L_R = \max(0, 1 - 3d_3/d_2)\overline{R}, \qquad U_R = (1 + 3d_3/d_2)\overline{R}, \tag{22.1}$$

$$L_S = \max\left(0, 1 - \frac{3\sqrt{1 - c_4^2}}{c_4}\right)\overline{s}, \qquad U_S = \left(1 + \frac{3\sqrt{1 - c_4^2}}{c_4}\right)\overline{s}. \tag{22.2}$$

These definitions agree with the NIST treatment [1]. They presume the subgroup and variance model used to derive the constants; normal-theory constants alone do not establish robustness to serial dependence or a highly nonnormal within-subgroup distribution.

A precision check. For a moving range of two independent observations, $d_2(2) = 2/\sqrt{\pi} = 1.1283791671$ and $d_3(2) = \sqrt{2 - 4/\pi} = 0.8525024664$. Consequently $D_4(2) = 1 + 3d_3/d_2 = 3.2665319193$. The full-precision five-observation constants give $D_4(5) = 2.1144991451$, whereas the NIST table prints 2.115. The latter is not the ordinary three-decimal rounding of the former. Both values are retained with their provenance. Changing a correct engine constant to reproduce a printed table's last digit would make the calculation less accurate.

The evidence must therefore test full-precision constants separately from their printed counterparts. Attaching the NIST citation to an existing hand-calculated case does not turn that case into a published numerical dataset. Limits, estimators and signal identities each need their own mapping.

22.3.2 Nelson special-cause rules (issue 57)

Nelson's Figure 1 defines eight supplementary patterns [2, p. 238]. These rules are applied to standardized chart positions, with the chart-specific centre and standard deviation determining the zone boundaries. Nelson's nine-point same-side rule is a different convention from the eight-point rule commonly associated with WECO. A list of four enabled rules is not, by its length alone, a WECO preset.

The retained default enables Nelson tests 1–4. Tests 5–6 are available for earlier warning and tests 7–8 for the diagnostic patterns described in the paper. This follows the routine and optional use distinguished in Figure 2, notes 4–6. A WECO preset would be a separate named option; this audit retains the Nelson convention.

Rule 8 requires eight consecutive observations outside the central one-sigma zone *with observations on both sides of the centre line*. Testing only $|z_i| > 1$ omits that last condition. A sequence of eight $z_i = 1.5$ values satisfies the old predicate but does not satisfy Nelson's rule 8. Alternating $-1.5, +1.5$ values do. A central-zone observation interrupts the window, and the membership of the complete eight-observation window must be examined; a sign change outside that window does not qualify it.

This is a correction to a decision rule, not a new statistical option. The retained regression cases distinguish one-sided windows, two-sided windows, central-zone interruptions and zone-boundary values. Because a recorded analysis can contain the enabled rules, the change also requires an explicit saved-analysis convention. The execution register distinguishes corrected new analyses from any intentional compatibility behavior when older records are replayed.

22.3.3 Binomial and Poisson attribute charts (issue 58)

For counts D_i of nonconforming units among n_i inspected units, the pooled binomial estimate and three-sigma p limits are

$$\overline{p} = \frac{\sum_i D_i}{\sum_i n_i}, \qquad \overline{p} \pm 3\sqrt{\frac{\overline{p}(1-\overline{p})}{n_i}}.$$

The limits are clipped to $[0, 1]$. Averaging the subgroup proportions without their denominators is generally a different estimator. An *np* chart uses the count scale with a common inspection size; a *u* chart uses defects per unit of exposure and accommodates varying exposure. These transformations must preserve their own units.

For Poisson defect counts at constant exposure, a *c* chart uses $\overline{c} \pm 3\sqrt{\overline{c}}$, with a nonnegative lower limit. The NIST wafer example has 25 counts totaling 400, hence $\overline{c} = 16$, $L = 4$ and $U = 28$ [3]. Wafer 24 has 31 defects and is beyond the upper limit. A count of 28 is on the limit and is not a beyond-limit signal under the strict comparison. This example checks the centre, limits and point classification together.

The Poisson example supplies published numerical evidence for the *c* chart. An *np* or *u* case derived from the equations is identified as an analytic extension, not as another observation from that published example. The three-sigma construction is a normal approximation to a discrete model; its actual tail probability depends on the mean and on integer rounding.

22.3.4 Tabular CUSUM (issue 59)

For target T, reference value $K = k\sigma$ and decision interval $H = h\sigma$, the zero-start two-sided tabular CUSUM is

$$C_i^+ = \max\{0, C_{i-1}^+ + x_i - T - K\}, \tag{22.3}$$
$$C_i^- = \max\{0, C_{i-1}^- + T - x_i - K\}, \qquad C_0^+ = C_0^- = 0. \tag{22.4}$$

Signals occur when either statistic is strictly greater than H. The audited convention continues the recursion after a signal; it does not reset the statistic automatically. Head starts and fast initial response are separate designs [4].

The commonly used $k = 0.5$, $h = 5$ choice is interpreted on the standardized individual-observation scale. With $T = 0$, $\sigma = 1$ and repeated $x_i = 1$, C_i^+ rises by 0.5 each step. It equals 5 at observation 10 and first exceeds the boundary at observation 11. This constructed example is useful for checking the recursion, units and strict boundary convention.

A quoted in-control average run length of approximately 465 describes the specified design under independent normal observations and known process parameters. It is not a universal guarantee for a chart whose target and scale are estimated from a short baseline, or for data with serial dependence. Neither a successful recurrence fixture nor that published average run length qualifies every practical CUSUM configuration.

22.3.5 Exponentially weighted moving averages (issue 60)

With $z_0 = T$ and $0 < \lambda \leq 1$, the EWMA is

$$z_i = \lambda x_i + (1 - \lambda) z_{i-1}.$$

Under independent observations with known variance σ^2, its startup variance is

$$\mathrm{Var}(z_i) = \sigma^2 \frac{\lambda}{2 - \lambda}\{1 - (1 - \lambda)^{2i}\}.$$

Time-dependent limits use this variance. Asymptotic limits omit the braced factor. These are two legitimate but different startup conventions; a published table using one cannot directly certify the other's early limits [5; 6, Sec. 9.2]. NIST independently states the recursion and asymptotic variance [29]; the complete paper and textbook support the finite-time factor used here.

Lucas and Saccucci's Table 1 uses $\lambda = 0.25$, $T = 0$, $\sigma = 1$ and asymptotic limits $\pm 3\sqrt{0.25/1.75} = \pm 1.1338934$. The first observations $1, -0.5, 0, -0.8$ give $z_1 = 0.25$, $z_2 = 0.0625$, $z_3 = 0.046875$ and $z_4 = -0.16484375$. On the full 19-observation sequence, the no-reset recursion signals at observations 16–19. A comparison using time-dependent limits may share those late signals while still differing at startup.

Montgomery's Example 9.2 and Table 9.10 provide a separate known-parameter example: $T = 10$, $\sigma = 1$, $\lambda = 0.1$ and $L = 2.7$, with signals at observations 29 and 30. These parameter values belong to the example. They do not replace the application's default $\lambda = 0.2$, $L = 3$. Likewise, a tabulated design with $L = 2.962$ and nominal average run length 500 is evidence for its stated design, not for every chart with a nearby multiplier.

22.3.6 Standardized short-run Z–MR charts (issue 66)

For supplied part-specific means μ_j and standard deviations σ_j, the plotted observation is $Z_i = (X_i - \mu_j)/\sigma_j$. The standardized individual chart has centre 0 and limits ± 3. For consecutive

standardized observations, $MR_i = |Z_i - Z_{i-1}|$. Under the independent standard-normal reference model,

$$E(MR_i) = d_2(2), \quad L_{MR} = 0, \quad U_{MR} = d_2(2) + 3d_3(2) = 3.6858865664.$$

Minitab's displayed values 1.128 and 3.686 agree at their stated precision [7]. A separately printed D_4 value is not a reason to replace the defining normal-range constants.

The supplied part parameters make this a known-parameter standardization case. It does not validate a method for estimating those parameters, an automatic choice of part groups, or an unannounced break in moving-range adjacency. The retained case must make the order and part assignments explicit.

22.3.7 Rare-event g and t charts (issue 67)

The statistical convention must specify whether the event itself is counted. For the audited g chart, $G \in \{1, 2, ...\}$ and $P(G = g) = p(1-p)^{g-1}$. The likelihood estimate is $\hat{p} = 1/\overline{G}$ and the displayed centre is $\overline{G}$. Probability limits use the geometric quantiles at 0.00135 and 0.99865, with the smallest support value whose cumulative probability reaches the requested level. The strict beyond-limit rule is retained. Discreteness prevents these limits from having exactly equal nominal tail probabilities for every value of p.

For exponential waiting times, the likelihood estimate of the mean is $\hat{\theta} = \bar{t}$, the centre is $\bar{t}$, and quantiles are $-\hat{\theta}\log(1-q)$. The exponential median is $\hat{\theta}\log 2$; choosing it as the centre would be a different display convention. The present mean-centred probability-limit construction is valid when named explicitly. It is not described as Benneyan's three-sigma chart or as an exact reproduction of a median-centred chart.

Benneyan's numerical example counts opportunities *before* the event, on support $\{0, 1, ...\}$, with mean 41.2 [8, p. 311]. Therefore $\hat{p} = 1/(41.2 + 1) = 0.02369668$. Adding one to each observation expresses the same process on the application's support and changes the mean to 42.2. It does not change p. The translated probability limits are 1 and 276, whereas the translated three-sigma upper limit is approximately 167.291. These quantities differ because the constructions differ. Using 41.2 as the mean on both supports would instead change the fitted probability.

22.3.8 Laney dispersion adjustment (issue 72)

For a p' chart, standardize the observed proportions using the binomial model and estimate their residual dispersion from consecutive differences:

$$z_i = \frac{p_i - \overline{p}}{\sqrt{\overline{p}(1-\overline{p})/n_i}}, \qquad \hat{\sigma}_z = \frac{\text{mean}\,|z_i - z_{i-1}|}{d_2(2)}.$$

The adjusted limits are $\overline{p} \pm 3\hat{\sigma}_z\sqrt{\overline{p}(1-\overline{p})/n_i}$, clipped to the probability scale. The u' construction uses the analogous Poisson standardization [9, pp. 533–535]. The valid adjacent pairs are part of the definition: filtering values must not silently join observations that were not consecutive in the chart's input order.

A positive $\hat{\sigma}_z < 1$ represents underdispersion and is retained. Zero, undefined and unestimable dispersion require an explicit result or refusal. They do not justify substituting $\hat{\sigma}_z = 1$. For example, repeated counts of 50 from groups of 100 yield identical standardized values and a moving-range estimate of zero. Replacing zero with one silently returns ordinary p-chart limits, changing the fitted model precisely where the estimate is degenerate.

The correction makes that condition visible. The regression evidence covers positive underdispersion, zero dispersion and inadequate valid adjacency, as well as ordinary overdispersion.

Any deliberately retained legacy replay behavior is identified separately from the rule for a new analysis.

22.3.9 Multivariate EWMA (issue 73)

For a vector X_i with reference mean μ and covariance Σ, the scalar-weight MEWMA starts at $Z_0 = \mu$ and uses

$$Z_i = \lambda X_i + (1 - \lambda) Z_{i-1}, \tag{22.5}$$

$$\Sigma_{Z_i} = \frac{\lambda}{2 - \lambda}\{1 - (1 - \lambda)^{2i}\}\Sigma, \tag{22.6}$$

$$T_i^2 = (Z_i - \mu)^\top \Sigma_{Z_i}^{-1} (Z_i - \mu). \tag{22.7}$$

The first monitored point is $i = 1$. Rows used solely to estimate a baseline do not advance the monitoring recursion. A singular covariance requires explicit handling; the quadratic form cannot silently become a different dimension or statistic.

Lowry and colleagues' Table 4 uses two variables, $\Sigma = \left(\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 1 \end{smallmatrix}\right)$, $\mu = 0$, $\lambda = 0.1$ and boundary $h = 8.79$ [10]. A constructed baseline with exactly that sample mean and covariance can exercise an API that estimates its baseline. Such rows are labelled constructed; they are not observations from the paper.

Table 22.6. Selected Lowry Table 4 comparisons. Recomputed values use the printed input coordinates; the printed statistic remains a separate field.

Observation	X_{i1}	X_{i2}	Recomputed T_i^2	Printed T_i^2
1	−1.19	0.59	3.288400	3.29
2	0.12	0.90	3.177196	3.17
9	1.56	1.58	9.644983	9.65
10	1.46	3.05	17.207445	17.21

Observations 9 and 10 exceed 8.79 when monitoring continues without reset. The discrepancies at observations 2 and 9 cannot both be dismissed as ordinary two-decimal rounding of the recomputation. Retaining the printed and derived values prevents that source-precision issue from being hidden inside a broad tolerance. The known-covariance design's average run length is also distinct from performance with an estimated covariance.

The indexed case compares all ten public T_i^2 outputs and the signal identities. The companion reference data retain all ten derived Z_i vectors, but the public result does not expose those vectors for a separate elementwise production comparison. Their presence in the reference data must not be counted as an additional executed output check.

22.3.10 Subgroup Hotelling and generalized variance (issue 74)

For m reference subgroups of equal size n and p variables, the audited Phase II subgroup-mean Hotelling boundary is

$$U = \frac{p(m+1)(n-1)}{m(n-1) - p + 1} F_{1-\alpha;\, p,\, m(n-1)-p+1}.$$

This construction uses the pooled within-subgroup covariance and its $m(n-1)$ residual degrees of freedom. It must not be exchanged with an individual-observation covariance or a Phase I formula solely because all are called Hotelling charts [6, Chap. 11; 11].

With $p = 2$, $m = 20$, $n = 10$ and $\alpha = 0.001$, the equation gives $U = 15.16503018$. The textbook prints 15.16; the recomputation rounds to 15.17 at two decimals. The fixture preserves the printed value as well as the separately derived expectation.

The generalized variance is the determinant of each subgroup covariance. Writing $\nu = n - 1$, its normal-theory constants are

$$b_1 = \nu^{-p} \prod_{i=1}^{p} (n - i), \tag{22.8}$$

$$b_2 = \nu^{-2p} \prod_{i=1}^{p} (n - i) \left\{ \prod_{j=1}^{p} (n - j + 2) - \prod_{j=1}^{p} (n - j) \right\}. \tag{22.9}$$

Then $E|S_i| = b_1|\Sigma|$ and $\operatorname{Var}|S_i| = b_2|\Sigma|^2$. The audited estimator of $|\Sigma|$ is $\overline{|S_i|}/b_1$. It is generally different from the determinant of the pooled covariance. Three-sigma limits consequently have centre $\overline{|S_i|}$ and multipliers $\max(0, 1 - 3\sqrt{b_2}/b_1)$ and $1 + 3\sqrt{b_2}/b_1$.

The determinant chart needs enough observations for a nonzero subgroup determinant, ordinarily $n > p$. A pooled mean chart can have different rank conditions. A fixture reconstructed from printed summary matrices is marked constructed and does not claim access to unpublished raw observations.

22.3.11 Moving-average charts (issue 75)

For width w and $k_i = \min(i, w)$ within the current monitoring segment, the expanding-startup moving average and limits are

$$M_i = \frac{1}{k_i} \sum_{j=i-k_i+1}^{i} x_j, \qquad T \pm \frac{3\sigma}{\sqrt{k_i}}.$$

Successive plotted averages share observations and are correlated even when the original data are independent. A three-sigma marginal limit does not therefore imply the independent-point run-length behavior of a Shewhart chart [6, Sec. 9.3].

In Montgomery's Example 9.3, $T = 10$, $\sigma = 1$ and $w = 5$. The first averages are 9.45, 8.72, 8.91, 9.5975 and 10.11. The full-window limits are 8.65835921 and 11.34164079; the largest average in the 30-point example is 11.17, so none is beyond its applicable limits. The example is a known-parameter check. It does not validate an estimator of σ based on moving ranges.

An option that withholds limits until a full window is available has a different startup display. It is also different from initially displaying ordinary individual-observation Shewhart limits. The manual and reference case must name the selected convention, including restarts at segment boundaries, instead of combining these alternatives under one description.

22.4 Measurement systems, capability and optimization

The reference comparisons in this section keep process variation, measurement variation, specification limits and uncertainty calculations distinct. A familiar index formula can be correct while the estimator or degrees of freedom used with it are different. The retained evidence must therefore identify both the statistic and the variation estimate it consumes.

22.4.1 Normal tolerance intervals and Howe's approximation (issue 61)

A two-sided normal tolerance interval $\bar{x} \pm ks$ aims to contain at least a specified population fraction P with confidence C. This differs from a confidence interval for the population mean

and from a prediction interval for one future observation [30]. For an independent normal sample of size N, s is the sample standard deviation and $\nu = N - 1$. The audited simple Howe approximation is

$$k_{\mathrm{HE2}} = z_{(1+P)/2}\sqrt{\frac{\nu(1 + 1/N)}{\chi^2_{1-C;\nu}}}. \tag{22.10}$$

The lower chi-square quantile is essential. A substitution of the upper quantile would reverse the intended confidence adjustment.

Howe gives this simple form in equation (2.13) and develops improved approximations thereafter [12]. The label *HE2* identifies the simple form in the independent R package `tolerance`; the package's `HE` option is a different calculation. The implementation retains HE2 and names it explicitly. Describing it as uniformly conservative is not supported by the paper.

Table 22.7. Howe Table 1, $N = 13$, $\nu = 12$, $P = C = 0.95$. Printed achieved confidence is a source result, not a new coverage simulation.

Construction	Printed factor	Printed achieved confidence
Simple form, λ_1	3.082	0.9486
Wald–Wolfowitz comparison, λ_2	3.081	0.9485
Improved approximation, λ_3	3.101	0.9510
Numerical reference factor	3.093	—

Equation (22.10) gives 3.0820886613. For these settings the improved approximation gives 3.1007994689. The distinction is numerically small but substantively important: the simple form's printed achieved confidence is below 0.95. The paper's separate sample-size and variance-degrees-of-freedom parameters also matter. Only rows representable by the product's $\nu = N - 1$ model are direct product fixtures; other rows can check a formula without pretending the application supports an independent externally supplied variance estimate.

For a one-sided normal tolerance bound the factor is instead obtained from a noncentral t quantile,

$$k_{\mathrm{one}} = \frac{t_{C;\nu,\sqrt{N}z_P}}{\sqrt{N}}.$$

The oracle calls therefore specify side, P, $1 - C$, N, ν and the two-sided method explicitly [13]. Agreement with an unspecified package default would not identify which interval was checked.

22.4.2 Weighted desirability and zero responses (issue 64)

Response desirabilities $d_j \in [0, 1]$ are combined using positive importance weights w_j:

$$D = \left(\prod_j d_j^{w_j}\right)^{1/\sum_j w_j}.$$

The audited individual desirability functions are piecewise linear within the specified lower, target and upper values; their shape exponents are one. Other response shapes are possible in the general framework but are not implied by validating this implementation [14; 15].

For positive weights, one zero component makes $D = 0$. Logarithms are useful for evaluating positive products stably, but replacing a zero with an arbitrary small positive constant changes the objective. With $(w_1, w_2) = (0.01, 1)$, an infeasible response pair $(d_1, d_2) = (0, 1)$ must have

desirability zero. Replacing the zero by 10^{-12} gives $10^{-12(0.01)/1.01} = 0.76065569$. A feasible pair $(0.5, 0.5)$ has desirability 0.5 and would then be ranked below the infeasible candidate. This example demonstrates an optimizer decision error, not merely a last-digit discrepancy.

Table 22.8. Derringer's Figure 1: three responses with importance weights $(1, 2, 4)$. Individual desirabilities are printed inputs.

d_1	d_2	d_3	Printed composite desirability
0.40	0.10	0.15	0.15
1.00	0.97	0.00	0.00
0.07	0.87	0.05	0.12
0.07	0.97	0.47	0.44

The retained checks cover exact zero propagation and candidate ranking, as well as the published rounded composites. If every explored candidate has zero desirability, the result states that condition. It must not report an artificial positive optimum created by a numerical floor. A legacy saved-optimization convention, where retained for reproducibility, is reported separately from corrected new optimization results.

22.4.3 Capability-index uncertainty (issue 65)

For specification limits L and U, the normal-theory index is

$$\widehat{C}_{pk} = \min\left\{\frac{U - \bar{x}}{3\hat{\sigma}}, \frac{\bar{x} - L}{3\hat{\sigma}}\right\}.$$

The two arguments of the minimum are $\widehat{C}_{pu}$ and $\widehat{C}_{pl}$; the uncentred potential-capability index is $\widehat{C}_p = (U-L)/(6\hat{\sigma})$ [31]. Whether the denominator describes overall or within-subgroup variation is part of the index definition. The same algebra with overall sample variation is ordinarily labelled P_{pk} in the application's terminology. Bissell's approximate uncertainty calculation uses

$$\mathrm{SE}(\widehat{C}_{pk}) \simeq \sqrt{\frac{(\widehat{C}_{pk})^2}{2\nu} + \frac{1}{9N}}, \tag{22.11}$$

with the effective degrees of freedom of the scale estimator [16, equations and example on pp. 333–339]. It is an approximate normal interval, not an exact confidence construction for every distribution or centring condition. NIST recommends at least 25 observations, preferably 100, for the common approximation. A small numerical example checks its implementation without establishing small-sample coverage.

For the overall sample standard deviation, $\nu = N - 1$. For a pooled within-subgroup standard deviation, $\nu = \sum_i (n_i - 1)$ under the common normal-variance model. For g equal-size subgroups of size n, the audited effective degrees of freedom for range and mean-standard-deviation estimators are

$$\nu_R = \frac{g}{2(d_3/d_2)^2}, \qquad \nu_S = \frac{g}{2(1/c_4^2 - 1)}.$$

These are estimator-specific approximations. Treating all four estimators as if they had $N - 1$ degrees of freedom would understate or distort their uncertainty.

The retained Bissell cases compare the public index and normal-interval endpoints, and the reported effective-df description. The independent reference also retains the standard error used

to construct each endpoint. The public result has no separate SE field; this audit therefore checks its effect on the interval rather than counting an unreturned SE as an additional production output.

Bissell's Table 3 supplies 75 measurements in 15 subgroups of five, with $L = 5.49$, $U = 5.79$ and overall mean 5.6136. Table 4 compares the four estimators. The original table is retained at its printed precision.

Table 22.9. Selected entries from Bissell Table 4. The paper uses approximately ± 2 SE for its displayed intervals.

Scale estimate	ν	Index	SE	Printed interval
Overall sample SD	74	0.853	0.080	[0.693, 1.013]
Mean range	54	0.831	0.089	[0.653, 1.008]
Mean subgroup SD	57	0.831	0.087	[0.657, 1.005]
Pooled subgroup SD	60	0.853	0.086	[0.679, 1.027]

The application uses the requested normal quantile, approximately 1.959964 at 95%, rather than fixing the multiplier to 2. Its interval endpoints therefore need a separate derived expectation. Some printed point and standard-error entries also differ in the last displayed digit from recomputation using the printed data. These are recorded as source discrepancies, not silently relabelled ordinary rounding. Unequal-subgroup and moving-range extensions require their own derivation and cases.

22.4.4 Single-sampling operating-characteristic curves (issue 69)

A binomial single-sampling plan inspects n units and accepts a lot when the number of nonconforming units is at most c. Its operating-characteristic curve is

$$P_a(p) = P(D \le c) = \sum_{d=0}^{c} \binom{n}{d} p^d (1-p)^{n-d}.$$

The inclusive endpoint is essential: accepting $D < c$ would be a different plan. The binomial model assumes independent unit outcomes at the stated fraction nonconforming, or an appropriate large-lot approximation [17].

For the NIST example $n = 52$, $c = 3$, the page prints acceptance probabilities 0.998, 0.930 and 0.739 at $p = 0.01$, 0.03 and 0.05, respectively. The first two agree with direct binomial evaluation at the printed precision. At $p = 0.05$, direct evaluation gives 0.7383174108, which rounds to 0.738, not 0.739. The printed 0.739 is preserved as a source discrepancy; the full-precision binomial calculation is retained as the independent analytic expectation. The tolerance is not enlarged to make both appear identical.

This comparison checks the binomial OC function. Finite-lot hypergeometric plans, risk-constrained plan searches and other sampling models are separate claims and require separate evidence.

22.4.5 Type 1 gage capability and bias (issue 70)

Let $T = U - L$ be the specification width, s the sample standard deviation of repeated measurements of one reference item, $b = \bar{x} - x_{\mathrm{ref}}$ the estimated bias, $SV = 6s$ the study variation and K

the allowed percentage of tolerance. The audited indices are

$$C_g = \frac{(K/100)T}{SV}, \qquad C_{gk} = \frac{(K/200)T - |b|}{SV/2}.$$

The Minitab methods page presents these formulas as equation images; those images were inspected directly [18]. Using their displayed definitions avoids confusing the tolerance percentage with a fraction, or a six-standard-deviation width with a half-width.

For $K = 20$, $T = 30$, $s = 1$ and $|b| = 0.2$, the values are $C_g = 1$ and $C_{gk} = 0.9333333333$. The repeatability percentage is $100SV/T = 20\%$. Bias inference uses $t = b/(s/\sqrt{n})$ with $n - 1$ degrees of freedom and a two-sided probability under the normal repeated-measurement model. A fixture must identify the number of repeats to validate that inference; the index calculation alone does not determine its p-value.

This study characterizes repeated measurements of the supplied reference item. It does not establish operator reproducibility, performance throughout the instrument's range, or stability across time.

22.4.6 Deming regression and jackknife intervals (issue 71)

Deming regression accounts for error in both measured variables. The ratio convention is explicit: $\lambda_T = \mathrm{Var}(\varepsilon_Y)/\mathrm{Var}(\varepsilon_X)$ in Tabelz. With centred cross-products S_{xx}, S_{yy} and S_{xy}, define $D = S_{yy} - \lambda_T S_{xx}$. For a nonzero cross-product, the fitted slope and intercept are

$$\hat{\beta} = \frac{D + \sqrt{D^2 + 4\lambda_T S_{xy}^2}}{2S_{xy}}, \qquad \hat{\alpha} = \overline{y} - \hat{\beta}\overline{x}.$$

An algebraically equivalent evaluation can improve numerical stability. The fitted latent abscissa for an observed pair is

$$\hat{x}_i = \frac{\lambda_T x_i + \hat{\beta}(y_i - \hat{\alpha})}{\lambda_T + \hat{\beta}^2}.$$

The NCSS treatment and the independent `mcr` implementation supply reference definitions and comparisons [19; 20].

The package uses the reciprocal error-variance convention. Consequently its input is `error.ratio=1/lambda_T`. A unit-ratio example cannot detect an inversion error. The retained ten-pair case `STAT-MSA-DEMING-007` uses $\lambda_T = 0.25$, corresponding to a reference ratio of 4; it is reused rather than duplicated under a new ID. Its ten pairs originate in the NCSS example and are also reproduced by NIST Dataplot. The pinned `mcr` reference gives slope 1.0011942278 and intercept -0.0897448990. The respective jackknife standard errors are 0.1871770528 and 1.7219874128; the 95% intervals are $[0.5695631700, 1.4328252856]$ and $[-4.0606549937, 3.8811651957]$. These full-precision package results are separate from the publication's displayed precision.

For the jackknife interval, let $\hat{\theta}_{(-i)}$ denote a leave-one-out coefficient estimate. The standard error is

$$s_{\mathrm{JK}}^2 = \frac{n-1}{n}\sum_i (\hat{\theta}_{(-i)} - \overline{\hat{\theta}}_{(-)})^2.$$

The audited interval centres on the full-sample estimate and uses $t_{1-\alpha/2;n-2}s_{\mathrm{JK}}$. Matching the coefficient alone does not verify the centring, degrees of freedom or interval construction. The pinned oracle call specifies Deming regression and jackknife intervals. Minimum-sample refusals, degenerate leave-one-out fits and preservation of the input row identity remain explicit application checks; a package comparison for successful coefficients does not cover those contracts.

22.5 Distribution diagnostics and simultaneous inference

The reference audit distinguishes a statistic from its calibration. An unchanged statistic can produce a different inference when a fitted-parameter correction, tail approximation or simultaneous critical value changes.

22.5.1 Grubbs' single-outlier test (issue 62)

For N independent observations from a normal population, the two-sided Grubbs statistic is

$$G = \frac{\max_i |x_i - \overline{x}|}{s},$$

where s is the sample standard deviation with divisor $N - 1$. Let $t_* = t_{1-\alpha/(2N);N-2}$. The critical value is

$$G_* = \frac{N-1}{\sqrt{N}} \sqrt{\frac{t_*^2}{N - 2 + t_*^2}}.$$

Both the two tails and the selection of the most extreme observation enter the critical probability [21]. Using a conventional $t_{1-\alpha/2}$ cutoff would omit the selection adjustment.

Inverting the statistic gives

$$t_G = \sqrt{\frac{N(N-2)G^2}{(N-1)^2 - NG^2}}, \qquad p = \min\{1,\, 2N[1 - F_{t;N-2}(t_G)]\},$$

with explicit limiting treatment when the denominator vanishes. The comparison must check the selected observation, statistic, critical value and two-sided probability. A constant sample has no positive sample scale and cannot supply an ordinary finite standardized statistic.

The retained NIST dataset contains

$$199.31, 199.53, 200.19, 200.82, 201.92, 201.95, 202.18, 245.57.$$

The candidate is 245.57. The independently derived reference gives $G = 2.4687646112$, critical value 2.1266450872 at $\alpha = 0.05$, and $p = 3.0026386821 \times 10^{-7}$. The case `STAT-HARV-NIST-GRUBBS-001` retains high-precision analytic expectations for those published inputs; those extra digits are not attributed to the NIST printed example. NIST's worked example asks the one-sided question of whether the largest observation is an outlier and prints a critical value of 2.032. The retained case adapts the published observations to the application's two-sided test, using NIST's general two-sided formula. It therefore does not use 2.032 as its expected cutoff.

The test addresses one candidate outlier under a normal reference model. Repeatedly deleting the selected value and applying the same nominal test is a different multiple-outlier procedure. Neither that procedure nor robustness to heavy tails follows from the present fixture.

22.5.2 Normal Anderson–Darling calibration (issue 63)

For ordered data $x_{(i)}$ and fitted normal cumulative probabilities F_i, the unadjusted Anderson–Darling statistic is

$$A^2 = -N - \frac{1}{N} \sum_{i=1}^{N} (2i - 1)\{\log F_i + \log[1 - F_{N+1-i}]\}.$$

The audited normal fit uses the sample mean and sample standard deviation. Stable log-probabilities are important in extreme tails; replacing a tail probability by zero would make the statistic infinite through numerical underflow rather than through the mathematical definition.

Two finite-sample corrections must be kept distinct. Stephens (1974) uses $1 + 4/N - 25/N^2$ for the relevant fitted-normal calibration [22]. The retained application convention uses the later factor $1 + 0.75/N + 2.25/N^2$ with its associated piecewise p-value approximation. The latter factor is independently inspectable in `nortest::ad.test` [23]. The later Stephens chapter cited by that package was not among the primary pages directly inspected in this audit. The evidence therefore identifies the package implementation as the inspected source for this later calibration; it does not assert direct verification of an unread 1986 table.

The 11-value example

$$148, 154, 158, 160, 161, 162, 166, 170, 182, 195, 236$$

gives $A^2 = 0.9467718796$. The 1974 factor produces 1.0954385384, slightly above its printed 1% critical value 1.092. The later factor gives 1.0289297700 and the retained approximation gives $p = 0.0104540240$. Thus the later method rejects at 5% but does *not* reject at 1%. Borrowing the earlier critical value while retaining the later adjustment would mix calibrations and could reverse a borderline decision.

The independent package comparison uses its supported sample-size domain (at least eight observations). Small-sample refusals, fitted-input handling and very extreme tails remain separately identified contracts. Agreement between two uses of the same approximation is not proof that its tail probabilities are exact for every sample size.

22.5.3 Johnson transformation parameterizations (issue 68)

Slifker and Shapiro describe three Johnson families [24, equation (1)]:

$$S_U: \quad z = \gamma + \eta \operatorname{asinh}((x-\epsilon)/\lambda), \qquad -\infty < x < \infty, \tag{22.12}$$

$$S_B: \quad z = \gamma + \eta \log \frac{x-\epsilon}{\epsilon+\lambda-x}, \qquad \epsilon < x < \epsilon+\lambda, \tag{22.13}$$

$$S_L: \quad z = \gamma + \eta \log((x-\epsilon)/\lambda), \qquad x > \epsilon, \tag{22.14}$$

with positive scale and shape parameters λ and η. For the first two families, the SciPy convention maps $(a, b, \text{loc}, \text{scale})$ to $(\gamma, \eta, \epsilon, \lambda)$. For a shifted lognormal fit with SciPy parameters $(s, \text{loc}, \text{scale})$, an equivalent form is $z = \gamma^* + \eta \log(x - \epsilon)$, where $\eta = 1/s$, $\epsilon = \text{loc}$ and $\gamma^* = -\log(\text{scale})/s$.

The paper's parameter estimates and chosen percentiles provide a useful transformation check. The S_U parameters are $(\gamma, \eta, \epsilon, \lambda) = (1.402, 2.333, 15.516, 1.585)$; the S_B parameters are $(2.373, 1.959, 0.295, 1.203)$. The example's percentile construction expects approximately $\pm 1, \pm 3$ in the first case and $\pm 0.5483, \pm 1.6449$ in the second.

Table 22.10. Johnson worked transformations using printed parameters. Agreement is assessed at the precision of those parameters.

Family	Printed input x values	Transformed values, approximately
S_U	16.689, 15.242, 13.581, 10.409	3.001, 1.001, −0.999, −2.999
S_B	0.786, 0.635, 0.516, 0.432	1.645, 0.548, −0.549, −1.646

The application fits a declared family by maximum likelihood. The paper's percentile-based family selection and parameter estimation are different procedures. Minitab's automatic selection and percentile estimation are also distinct from this declared-family workflow [32]. A fixture that supplies the printed parameters verifies the transformation and convention; it does not

independently qualify the application's maximum-likelihood fit or select its family. Similarly, a normality diagnostic calculated after fitting a transformation is not automatically a calibrated goodness-of-fit test for the whole fitted family.

22.5.4 Fieller intervals for inverse calibration (issue 76)

Let A and B be approximately jointly normal estimates with covariance entries V_{AA}, V_{AB} and V_{BB}. Fieller's confidence set for their ratio consists of all θ satisfying

$$(A - \theta B)^2 \leq q^2(V_{AA} - 2\theta V_{AB} + \theta^2 V_{BB}). \tag{22.15}$$

Here q is the reference quantile appropriate to the model. In the audited probit inverse-calibration calculation it is a normal quantile based on an asymptotic coefficient covariance, rather than an exact finite-sample Student construction [25].

For probit coefficients g_0, g_1, requested probability p, centre c and scale s, the inverse is

$$x_p = c + s\frac{\Phi^{-1}(p) - g_0}{g_1}.$$

Thus $A = \Phi^{-1}(p) - g_0$, $B = g_1$ and $V_{AB} = -\operatorname{Cov}(g_0, g_1)$. The covariance sign must change with the negated intercept. For $g_0 = -1.2$, $g_1 = 0.8$, $p = 0.5$, $V_{00} = 0.09$, $V_{11} = 0.04$ and $V_{01} = 0.012$, the ratio estimate is 1.5 and its 95% bounded interval is approximately $[0.644086, 3.493325]$ before the affine centre/scale transformation.

Expanding equation (22.15) gives a quadratic inequality. Its leading coefficient is $B^2 - q^2 V_{BB}$. With a positive coefficient and two real roots, the set is the bounded interval between those roots. A negative coefficient can instead give two disjoint rays or the entire real line. A zero leading coefficient gives a linear or constant inequality and must not be divided by as if it were a quadratic. The numerical details at that boundary are contract checks, not opportunities to return invented finite endpoints.

The application retains its bounded-interval interface. When a bounded interval cannot be reported, the visible result says *No bounded Fieller interval*. It discloses that the full set may have a shape the interface does not display. The existing machine status is preserved where required for compatibility; its name alone must not imply that every such set is the whole real line.

The pinned `twopartm::FiellerRatio` implementation supplies an independent bounded calculation and explicit examples of disjoint and whole-line sets [26]. Its behavior near exact equality of the leading coefficient is not treated as an authoritative numerical oracle where cancellation makes its own computation unstable. No unsupported case is silently dropped from the evidence count.

22.5.5 GT2 and Gabriel multiple comparisons (issue 39)

This part of the audit corrects the future F137 method contract. It is not evidence that the planned multiple-comparison feature is already implemented. The primary papers resolve the critical-value convention and define what a future implementation must reproduce.

Let $m_{c,\nu;1-\alpha}$ be the $(1-\alpha)$ quantile of the maximum absolute value of c independent standard-normal variables divided by an independent $\sqrt{\chi^2_\nu/\nu}$. This is the studentized maximum modulus (SMM) distribution. For k group means and all pairwise comparisons, the required index is $c = k(k-1)/2$. The index counts the comparisons; it is not replaced by k merely because there are k groups.

For independent normal samples with common variance estimate s^2, Hochberg's GT2 pairwise interval has half-width

$$m_{c,\nu;1-\alpha}\, s\sqrt{1/n_i + 1/n_j}.$$

Hochberg's inequality (4.4) supplies a useful bound by the corresponding Bonferroni t critical value [27]. A formerly proposed acceptance invariant that the GT2-to-Tukey width ratio always lies between 1.02 and 1.10 is false. For $k = 4$, $\nu = 100$ and $\alpha = 0.01$, independent numerical integration gives $m = 3.2294577401$ and studentized range $q = 4.5162582356$, so $\sqrt{2}m/q = 1.0112670040$. This counterexample is retained rather than enforcing the false invariant.

Gabriel constructs a group interval

$$[l_i, u_i] = \overline{y}_i \pm \frac{m_{c,\nu;1-\alpha} s}{\sqrt{2n_i}}, \tag{22.16}$$

then forms the interval for $\mu_i - \mu_j$ as $[l_i - u_j, u_i - l_j]$ [28, pp. 724–725]. For equal sample sizes this is identical to GT2 when both use the same SMM calibration. Replacing m by $q/\sqrt{2}$ is a distinct Tukey calibration and must be named as such. With unequal sizes Gabriel's construction has approximate simultaneous coverage and can be liberal. A sample-size-ratio heuristic or a favorable simulation in the paper is not a general proof of familywise error control.

Table 22.11. Gabriel's four-group example, $s = 2.699$, $\nu = 266$, $\alpha = 0.10$, $c = 6$. Group intervals recomputed with $m = 2.3916397674$ are shown alongside the printed intervals.

Group	n_i	Mean	Recomputed interval	Printed interval
1	56	−1.3571	[−1.967044, −0.747156]	[−1.967, −0.747]
2	61	0.4918	[−0.092611, 1.076211]	[−0.093, 1.076]
3	100	1.5900	[1.133560, 2.046440]	[1.134, 2.046]
4	53	2.0566	[1.429632, 2.683568]	[1.429, 2.684]

For example, the recomputed interval for $\mu_2 - \mu_1$ is [0.654545, 3.043255], and that for $\mu_4 - \mu_3$ is [−0.616808, 1.550008]. Gabriel's general contrast construction also yields [−0.131694, 2.163094] for $\mu_4 - (\mu_2 + \mu_3)/2$. The corresponding group-interval construction for the unweighted mean of groups 2 and 3 gives [0.520474, 1.561326]; the paper prints [0.521, 1.561]. Printed inputs, the paper's rounded $m = 2.392$, and recomputation with a full-precision critical value are kept as distinct evidence fields.

The corrected future contract reinstates Gabriel on this primary-source basis, retains Tukey-b as deferred, and removes the false ratio invariant. The normal-means model with an independent common variance estimate does not by itself authorize arbitrary generalized-linear-model coefficients or every repeated-measures covariance estimator. Broader covariance support requires a separately justified contract and reference evidence.

22.6 References and inspected source locations

The method sections identify the equations, tables and examples used. The source register below records the publication identity; the execution manifest records source hashes and the actual numerical-reference versions.

Page numbers below refer to the publication's printed pagination, not to a PDF viewer's page counter. Supplied copies can contain covers or form part of a larger anthology. Online methods pages were inspected on 27 September 2026. Package comparisons use the retained lockfile versions; a current web page is not substituted for that execution record.

[1] NIST/SEMATECH. *e-Handbook of Statistical Methods*, sections 6.3.2.1 and 6.3.2.2, control charts for subgroup means/dispersion and individual measurements. Definitions of A_2, D_3, D_4, c_4 and moving-range limits; printed constant tables. Subgroup charts and individual charts.

[2] Nelson, L. S. (1984). The Shewhart Control Chart—Tests for Special Causes. *Journal of Quality Technology*, 16(4), 237–239. Figure 1, p. 238: eight special-cause tests, including the both-sides condition for test 8. Supplied PDF: `The-Shewhart-Control-Chart-Tests-for-Special-Causes-Lloyd-Nelson-Journal-of-Quality-Technology.pdf`.

[3] NIST/SEMATECH. *e-Handbook of Statistical Methods*, sections 6.3.3.1 and 6.3.3.2. Poisson c and binomial p charts; 25-wafer example with total defect count 400. Poisson charts and binomial charts.

[4] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 6.3.2.3. Tabular CUSUM definitions, reference value and decision interval; CUSUM control charts.

[5] Lucas, J. M., and Saccucci, M. S. (1990). Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements. *Technometrics*, 32(1), 1–12. Table 1: zero-start EWMA example with asymptotic limits; Table 3: design constants under stated run-length conditions. Supplied complete PDF: `lucas1990_full.pdf`. The shorter `lucas1990.pdf` is incomplete and was not used as the complete article.

[6] Montgomery, D. C. (2009). *Introduction to Statistical Quality Control*, 6th ed. Wiley. Example 9.2 and Table 9.10, p. 421 (EWMA); Example 9.3 and Table 9.14, p. 429 (moving average); Section 11.3, pp. 501–505, equations (11.19)–(11.21) and Example 11.1 (multivariate charts and the Phase II subgroup example). Supplied PDF: `Montgomery_Statistical_Quality_Control.pdf`.

[7] Minitab. *Methods and formulas for Z–MR Chart*. Standardized individual values and moving-range centre/limits. Official methods page. Printed display constants are distinguished from full-precision normal-range constants in Section 22.3.1.

[8] Benneyan, J. C. (2001). Number-Between g-Type Statistical Quality Control Charts for Monitoring Adverse Events. *Health Care Management Science*, 4, 305–318. In particular p. 311: alternative control limits and the zero-based mean-41.2 example. Supplied PDF: `Number-Between_g-Type_Statistical_Quality_Control_.pdf`.

[9] Laney, D. B. (2002). Improved Control Charts for Attributes. *Quality Engineering*, 14(4), 531–537. Standardized proportions/rates and moving-range dispersion adjustment, pp. 533–535. Supplied PDF: `Laney_Improved_Control_Charts.pdf`.

[10] Lowry, C. A., Woodall, W. H., Champ, C. W., and Rigdon, S. E. (1992). A Multivariate Exponentially Weighted Moving Average Control Chart. *Technometrics*, 34(1), 46–53. Covariance of the MEWMA vector and Table 4's ten-observation bivariate example. Supplied PDF: `lowry1992_Multivariate_Exponentially_Weighted.pdf`.

[11] Alt, F. B., and Grimshaw, S. D. (2013). Multivariate Quality Control. In S. I. Gass and M. C. Fu (eds.), *Encyclopedia of Operations Research and Management Science*, pp. 1014–1017. Subgroup Hotelling monitoring and generalized-variance moments, especially pp. 1016–1017. Supplied anthology excerpt: `Multivariate_Process_Control.pdf`.

[12] Howe, W. G. (1969). Two-Sided Tolerance Limits for Normal Populations—Some Improvements. *Journal of the American Statistical Association*, 64(326), 610–620. Equations (2.13)–(2.14), p. 614; principal result, p. 616; Table 1, p. 619. Supplied PDF: `Two-Sided Tolerance Limits for Normal Populations Some Improvements.pdf`. The table separates sample size from variance degrees of freedom.

[13] R package `tolerance`, version 3.0.0, function `K.factor`. The retained generator explicitly selects `method="HE2"`, `side=2`, `alpha=1-C`, `P=P` and `f=N-1` for the audited two-sided interval. Package documentation; inspectable source mirror. The one-sided call and its noncentral-*t* expectation are retained separately.

[14] Derringer, G. C. (1994). A Balancing Act: Optimizing a Product's Properties. *Quality Progress*, 27(6), 51–58. Figure 1 (importance-weighted composites) and Figure 2 (two-response illustration). Supplied PDF: `279_-_A_Balancing_Act.pdf`.

[15] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 5.5.3.2.2, multiple-response optimization by desirability. Individual desirability functions and geometric combination.

[16] Bissell, A. F. (1990). How Reliable is Your Capability Index? *Applied Statistics*, 39(3), 331–340. Effective degrees of freedom and approximate standard errors; Table 3 (75 measurements) and Table 4 (indices, standard errors and intervals). Supplied PDF: `bissel.pdf`. This is an *Applied Statistics* article, not a *Journal of Quality Technology* article.

[17] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 6.2.3.2, single-sampling plans. Binomial operating-characteristic example with $n = 52$, $c = 3$. The $p = 0.05$ printed-table discrepancy is retained in the evidence.

[18] Minitab. *Methods and formulas for Type 1 Gage Study*. C_g, C_{gk}, repeatability and bias formulas, including the equation images for the two capability indices. Official methods page.

[19] NCSS. *Deming Regression*, procedure documentation. Error-variance-ratio convention, coefficient estimates and jackknife confidence intervals. Official procedure PDF.

[20] R package `mcr`, version 1.3.3.1. `mcreg(method.reg="Deming",method.ci="jackknife")`, with `error.ratio=1/lambda_Tabelz`; interval construction inspected in `mcCalcCI.r`. Package documentation; inspectable source mirror.

[21] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 1.3.5.17.1, Grubbs' test for outliers. Two-sided statistic, critical value and worked data example.

[22] Stephens, M. A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons. *Journal of the American Statistical Association*, 69(347), 730–737. Publisher DOI record. Fitted-normal statistic correction and the 11-observation example. Supplied PDF: `981993412-stephens 1974-1.pdf`.

[23] R package `nortest`, version 1.0-4, function `ad.test`. Inspected later fitted-normal adjustment $1 + 0.75/N + 2.25/N^2$ and piecewise probability approximation. Package documentation; inspectable source mirror. The package cites Stephens' 1986 chapter in D'Agostino and Stephens, *Goodness-of-Fit Techniques*; that chapter was not directly inspected for this audit.

[24] Slifker, J. F., and Shapiro, S. S. (1980). The Johnson System: Selection and Parameter Estimation. *Technometrics*, 22(2), 239–246. Equation (1) and worked S_U/S_B percentile examples. Supplied PDF: `slifker1980.pdf`.

[25] Franz, V. H. (2007). Ratios: A Short Guide to Confidence Limits and Proper Use. arXiv:0710.2024, especially pp. 7–8 on Fieller sets. This inspected exposition supports the set-shape distinction; the audit does not claim inspection of Fieller's original 1954 paper.

[26] R package `twopartm`, version 0.1.0, function `FiellerRatio`. Numerator/denominator order, covariance mapping and finite/disjoint/whole-line examples. Package documentation; inspectable source mirror.

[27] Hochberg, Y. (1974). Some Generalizations of the T-Method in Simultaneous Inference. *Journal of Multivariate Analysis*, 4, 224–234. SMM-based simultaneous intervals and inequality (4.4), p. 229. Supplied PDF: `hochrbeg.pdf`.

[28] Gabriel, K. R. (1978). A Simple Method of Multiple Comparisons of Means. *Journal of the American Statistical Association*, 73(364), 724–729. Definitions and contrast construction, p. 724; four-group example, p. 725; coverage discussion and simulations thereafter. JSTOR stable record 2286265. Supplied PDF: `gabriel1978.pdf`.

[29] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 6.3.2.4, EWMA control charts. Recursion, initialization and asymptotic variance. The finite-time variance is corroborated separately in the complete Lucas–Saccucci article.

[30] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 7.2.6.3, tolerance intervals for a normal distribution. Two-sided simple Howe approximation and exact one-sided noncentral-t construction, with content and confidence distinguished.

[31] NIST/SEMATECH. *e-Handbook of Statistical Methods*, section 6.1.6, process capability. Index definitions and the approximate C_{pk} interval using an ordinary sample standard deviation; sample-size guidance and the distinct chi-square interval for C_p.

[32] Minitab. *Methods and formulas for Johnson Transformation*. Family definitions, parameter notation and the product's separate percentile-fit/automatic-selection procedure. Official methods page.

CHAPTER 23

Validation: capability workflow

Protocol: **WF-CAPABILITY-001**, version 1.0.0.

This is executable engineering evidence for one bounded desktop workflow. It is not product-wide statistical qualification, regulatory certification, customer acceptance, or release approval.

23.1 Scope and intended use

The protocol imports a CSV process study, removes one incomplete record through the missing-values preparation UI, retains but excludes a synthetic quarantine record, applies a two-sided specification, explicitly selects the Normal model, runs capability, reviews the normality check and warning, saves a semantic report, and exports HTML. It saves and reopens the project with fresh controllers and caches, swaps the identity of the excluded row without changing the exclusion count, proves the saved report becomes stale without mutation, reopens that stale state, restores the intended population, and reruns the study.

The analyzed fixture has 20 values: eighteen `1` values followed by `20` and `100`, with LSL `0` and USL `120`. Its deliberate skew makes the explicit Normal model check fail, so the workflow also proves that HypoNum retains the selected model and surfaces the warning instead of silently choosing another model. The synthetic `MISSING` and `QUARANTINE` rows are not part of the numeric reference.

23.2 Execute

Run from the matching source checkout with locked dependencies:

```
uv run --locked python tools/run_capability_workflow.py --output-dir .tmp/capability-
    workflow-evidence
uv run --locked python -m pytest tests/validation/test_capability_workflow.py tests/
    validation/test_interpretation_correctness.py tests/unit/analysis/test_spec_sheet.
    py tests/unit/desktop/test_spec_capability_controller.py
```

The output directory must be new or empty. Existing evidence, including failures, is never overwritten. Exit 0 requires every mandatory check and numeric comparison to pass. Missing or duplicate checks, exceptions, input/specification changes, and source-tree changes during execution fail the protocol.

23.3 Independent numeric acceptance

The reference constants are fixed analytic results, not values produced by the implementation under test. For the 20 analyzed measurements, `n=20`, `sum=138`, and corrected sum of squares is `9465.8`. Overall sigma is therefore `sqrt(9465.8/19)`. Consecutive absolute moving ranges sum to `99`. The analytic reference uses `d2=2/sqrt(pi)`; production stores the declared method constant as the rounded `1.128379167`. The `1e-9` relative tolerance explicitly accommodates that approximately `8.5e-11` relative rounding difference. This is a check of the declared

HypoNum method policy, not an independent certification of its d2 choice. Cp/Cpl/Cpu/Cpk and Pp/Ppl/Ppu/Ppk then follow the formulas disclosed by the product. Normal tail probability supplies DPMO and Z.bench. The absolute tolerance is `1e-12`; counts are exact.

Focused validation tests separately cover:

- Normal two-sided and one-sided capability against fixed analytic constants;
- undefined indices and explanatory warnings for zero variance, one observation, and no finite observations;
- binomial Wilson 95% limits against fixed reference constants;
- OLS positive, negative, and non-significant slope conclusions, p-values, and 95% confidence intervals against a closed-form five-point design; and
- explicitly unavailable p-values and intervals, plus warnings, for a ridge fit.

The OLS fixture uses centered `x=(-2,-1,0,1,2)` and residual basis `(1,-2,2,-2,1)`. Thus `Sxx=10`, `SSE=14*s^2`, residual df is 3, `SE(slope)=s*sqrt(14/30)`, and the tabulated 95% critical value is `t(.975,3)=3.182446305284263`. For df=3 the test independently evaluates the closed-form CDF `1/2+[atan(u)+u/(1+u^2)]/pi`, where `u=|t|/sqrt(3)`, and checks the frozen two-sided p-values against it. Wilson limits are independently reconstructed from the fixed `z=1.959963984540054` score formula. No second regression or capability implementation is invoked as an oracle. The workflow's Shapiro p-value is only warning-path evidence (`p<.05`), not independent numerical certification of the Shapiro implementation.

23.4 Evidence and limitations

The runner retains `results.json`, `summary.md`, generated input and specification files, full initial/rerun capability payloads, saved report JSON, current/stale HTML exports, two project packages, and QML screenshots. Artifacts carry SHA-256 hashes. The protocol uses the real QML controls, desktop bridge/controller, threaded capability engine, report store, and project save/open path; it does not mock or inject analysis outputs.

This scope validates only the Normal continuous model in the end-to-end workflow. It does not validate lognormal, Weibull, Box-Cox, Poisson, other statistical methods, native mouse/keyboard behavior, DPI/accessibility, packaged candidates, or clean-machine installation. The focused binomial and Fit Y by X tests are method-level correctness checks, not additional end-to-end workflow claims. Offscreen screenshots are review aids rather than an accessibility audit.

CHAPTER 24

Validation: industrial ozone workflow

Protocol: **WF-IMPORT-REPORT-001**, version 1.0.0.

This is an executable engineering proof for one defined desktop workflow. It is not a product-wide qualification, regulatory certification, or release approval. Independent statistical/quality review and customer-specific acceptance remain open.

24.1 Intended use and scope

An analyst imports calibration measurements, reviews and removes duplicate and incomplete records, excludes a documented non-study observation, fits an ordinary least-squares calibration with an intercept, and retains an interpretable report. The project must preserve the prepared records, exclusions, report values and source-change warnings through repeated saves and reopening with fresh controllers and separate caches.

The reference is the archived **NIST Norris ozone-monitor calibration** dataset: 36 observations, response `y`, predictor `x`. The fixture appends three explicitly synthetic records: an exact duplicate, a missing response, and `QUARANTINE`, a non-study observation with response -999. These additions test preparation and population control; they are not part of the NIST reference.

Configuration: CSV, UTF-8, comma delimiter, decimal point, header on row 1; linear OLS with intercept, listwise deletion, no holdout or imputation, 95% confidence. Report exports covered by this protocol are HTML and editable LaTeX source.

24.2 Execute

Run from the matching source checkout using its locked dependencies:

```
uv run --locked python tools/run_industrial_workflow.py --output-dir .tmp/industrial-
    workflow-evidence
uv run --locked python -m pytest tests/validation/test_industrial_workflow.py tests/
    unit/desktop/test_fit_report.py tests/unit/analysis/test_fit_y_by_x.py tests/unit/
    desktop/test_result_reports.py
```

The output directory must be new or empty. Existing evidence, including failures, is never overwritten. Exit 0 requires every mandatory check and every numerical comparison to pass. Missing or duplicate checks, reference checksum failures, exceptions and source-tree changes during execution fail the protocol.

24.3 What the protocol demonstrates

Stage	Acceptance evidence
Reference and import	Checksummed archived original, normalized rows and expectations; 39 imported records

Stage	Acceptance evidence
Preparation	The duplicate and incomplete row are selected before removal; the exact expected sample sequence remains
Exclusion	QUARANTINE stays in the table; surviving source-row identities remain unchanged; the fit uses precisely the 36 reference pairs
Desktop analysis	Actual QML Fit model action and production engine; 15 independent comparisons; method settings and nonempty interpretations checked
Reporting	Actual Add to results action; numerical parameter table, confidence limits and interpretations retained; explanatory note and seven fit cards
Export	HTML contains the interpretations and has no initial source-change warning; LaTeX contains report tables; full payloads retained separately
First reopen	Exact prepared values, report catalog and excluded sample restored, with no false stale warning
Sorting and second reopen	Sorting retains row identity and the excluded sample; sorted order and immutable reports survive reopening
Same-count exclusion swap	Replace excluded QUARANTINE with N001; all newly saved fit cards become stale, live fit is invalidated, stored cards remain unchanged
Third reopen and rerun	Swapped exclusion and stale state survive reopening; restoring the intended population reproduces all 15 numerical references
Evidence integrity	Original CSV unchanged; source baseline unchanged throughout execution; output artifacts carry SHA-256 hashes

Automation invokes the real QML setup/run controls and menu signals, with the public desktop bridge for row selection, report creation and save/open. It does not inject analysis outputs or replace the analysis, importer, preparation, reporting or persistence implementations. It is **offscreen QML evidence**, not proof of physical mouse/keyboard interaction or a clean-machine installation.

24.4 Independent numerical acceptance

The existing NIST-LRE-1 requirements remain unchanged: intercept 12 relative decimal digits, slope 14, R-squared 15, and exactly 36 observations.

Newly covered certified outputs are the two coefficient standard errors, residual standard deviation, model/error sums of squares, model/error mean squares and F. Their engineering requirement, **WF-NORRIS-1**, was fixed at 10 relative decimal digits before execution. This is an engineering threshold pending independent review, not a claimed NIST or vendor acceptance standard. Degrees of freedom must equal 1, 34 and 35 exactly; the desktop represents them as floats.

Residual standard deviation is derived from the desktop ANOVA error mean square. It is not compared with the UI's RMSE: RMSE divides residual sum of squares by `n`, whereas residual standard deviation uses residual degrees of freedom.

Published expected values come from the archived Norris reference, not from the implementation under test. Engineering comparisons retain full precision and raw errors. Report tables intentionally use five significant digits; the protocol checks their displayed parameter values against the original desktop payload. Confidence limits, p-values and interpretation text are checked for preservation; this increment does not independently certify their numerical values.

24.5 Defects identified and changes made

- **WF-DEF-001: incomplete fit report.** Previously, Add to results retained the scatter plot and headline metrics only. It now atomically adds the model summary, settings/warnings, ANOVA with interpretations, parameter estimates with confidence limits and interpretations, and residual/Q-Q plots. Report-capacity rejection cannot leave a partial addition. Existing saved reports are not retroactively changed.
- **WF-DEF-002: intercept precision.** On the archived original row order, the uncentered desktop SVD fit returned -0.26232307377375008 against certified -0.262323073774029, narrowly missing the retained 12-digit requirement. Full-rank OLS is now solved after centering predictors and response, with coefficients and covariance transformed back to original units. Rank-deficient fits preserve the prior minimum-norm SVD convention. Original, ascending and descending reference orders have permanent numerical regression coverage. No threshold was relaxed.
- An existing noiseless linear-fixture test required a strictly positive confidence band width caused by floating-point roundoff. It now permits a collapsed band for an exact fit; noisy-data interval tests retain their positive-width requirements.

24.6 Evidence and limitations

Engineering execution on **2026-09-12 16:55 UTC** passed all **24 protocol checks** and all **15 numerical comparisons**, including a second comparison after reopening and restoring the intended population. The focused report/fit/protocol regression suite, the additional rank-deficiency regression, targeted strict type checks and lint checks passed. Expected diagnostics from noiseless/rank-deficient statistical fixtures do not constitute a general numerical-domain qualification.

The evidence run used an isolated copy of the working tree because unrelated files were changing during the earlier run; the earlier baseline-integrity failure was retained. Its source-tree SHA-256 is `2b6e2930251daae6478f018ff3077604ce96fcbfa15ef5c4413e4b2edda3cdaa`. The local retained snapshot is `.tmp/industrial-proof-snapshot-20260912`, with evidence under its `.tmp/evidence/` directory. The source snapshot predates this execution-note addition and has not been rewritten to match it.

The runner retains `results.json`, `summary.md`, the generated input, full desktop payloads, saved report JSON, HTML/LaTeX exports, three project packages and QML screenshots. It records source commit and working-tree hashes, dependency lock and installed versions, OS, architecture and Qt platform. Keep the complete evidence directory together with the matching checkout; `.tmp` is local working storage, not a controlled release archive.

QML warnings are retained in the report. Existing Excel-preview delegate and report dialog layout warnings observed during development are separate from numerical and persistence acceptance; a passing protocol is not a claim that all UI warnings have been resolved. Screenshots are review aids, not a complete accessibility audit.

Excluded: Excel import/export, PDF output, other statistical methods, native DPI and assistive technology, signed installers, external-source relinking, crash recovery, protected audit histories and complete operation-graph replay. These remain separate qualification scopes. Legacy desktop reports are immutable snapshots; this proof does not relabel them as versioned statistical receipts.

See also Chapter 21, Chapter 20, and Chapter 19.

CHAPTER 25

Validation: time-series workbench

Status: engineering validation record, 2026-09-13. This record covers the current univariate and advanced desktop feature set. It does not replace independent statistical review, native release acceptance, or an institution's model-risk approval.

25.1 Scope and contracts

The validated workflows include rolling univariate forecasting, dynamic regression, GARCH, VAR/VECM, Zivot-Andrews, STL/MSTL, saved specifications, replay, report adapters, CSV export, and the QML setup/results flow. Advanced results use schema `1.0.0`; the spawned time-series worker uses protocol `1.2.0`. Specifications retain estimation settings, units, calendars, time zones, random seeds, simulation counts, and dependency versions. Data fingerprints cover every selected time and series column. Replay rejects a changed fingerprint.

Rolling studies reserve a final evaluation period, repeat estimated transformations and automatic selection within each training window, preserve the requested sliding window for final and future fits, and retain failed validation origins. Saved-study comparison treats differing targets, units, horizons, rolling samples, or final evaluation samples as incompatible.

25.2 Numerical checks

The focused numerical suite includes the following independent or invariant checks:

Area	Check
Drift and transformed forecasts	Difference-plus-mean and level-drift point forecasts and joint multi-step interval widths agree within the declared tolerance.
Publication/vintage handling	Later revisions do not change as-of rolling or final forecasts; forecasts cross unreleased response periods before scoring.
GARCH scale	Decimal and percentage return fits differ by the required dimensional factors while persistence is invariant.
GARCH mean handling	Adding a constant to returns shifts the fitted conditional mean by the same amount while squared innovations, variance forecasts, the variance benchmark, and QLIKE remain invariant.
VAR/VECM lag convention	One selected level lag maps to zero lagged differences at rank zero; two level lags map to one lagged difference. The boundary model forecasts, diagnostics, and impulse-response uncertainty remain available.

Area	Check
Rank-zero VAR intervals	A long simulated bivariate random walk matches the analytic `z*sigma*sqrt(h)` level-forecast half-width. Marginal differenced endpoints are not accumulated.
Interval coverage	A fixed-seed 100,000-path Gaussian random-walk simulation verifies that the returned horizon-four interval covers at the configured 95% rate within ± 2 percentage points.
VAR/VECM output	Named-series rank-zero, full-rank, and cointegrated paths return forecasts, whiteness/stability diagnostics, and impulse-response uncertainty without losing names.
STL/MSTL	Effective period order is retained, unsupported boundary periods fail clearly, residual seasonality is reported, and endpoint revision sensitivity is measured.
Seasonal forecasting transform	STL is first fitted on the inner selection training split and refitted at every study window; future seasonal phase returns to original units.
Calendars	NYSE holidays and future sessions are recognized in `America/New_York`; irregular observations without an explicit calendar fail.
Saved-study comparison	Rolling and untouched-final evaluation samples are fingerprinted and checked separately; changing only the final holdout makes displayed final metrics incompatible.

All random checks use fixed seeds. The coverage test evaluates the implemented interval against an independently generated Gaussian error distribution rather than calculating its expectation through the forecasting implementation.

25.3 Execution

Run from the locked repository environment:

```
uv sync --locked --all-groups
uv run ruff check .
uv run mypy src/tabelz
uv run --locked python -m pytest -q --junitxml=.tmp/time-series-pytest.xml
```

Run `ruffformat--check` on every changed Python file. The repository has a separately tracked formatting backlog in unrelated files, so a repository-wide format result must be interpreted against the candidate's recorded baseline rather than reported as a new time-series failure.

The targeted gate is:

```
uv run --locked python -m pytest -q `
  tests/unit/analysis/test_time_series.py `
  tests/unit/analysis/test_time_series_advanced.py `
  tests/unit/analysis/test_time_series_workbench.py `
  tests/unit/desktop/test_time_series_controller.py `
  tests/integration/test_qml_bridge.py
```

Retain the JUnit file with the commit, `uv.lock` hash, platform, interpreter, and the dependency versions emitted by the result payload. A focused pass does not override a failed broader gate.

25.4 2026-09-13 local execution

The final Windows engineering run used the locked environment and the completed working tree based on commit `1bd6d7e`. Pytest collected 1,769 tests: 1,767 passed, two were skipped, and none failed or errored in 1,240.762 seconds. The JUnit artifact was `.tmp/time-series-workbench-pytest-final-2.xml` with SHA-256 `ae676b7bad3af1385c2e607c1a6e2626acca9f7c39ecb3ea2f7fd425f2f3e3ab`. Repository-wide Ruff lint and mypy checks also passed. The changed Python files passed Ruff's format check.

The first broader run exposed two release-gate regressions from the earlier workbench increment: optional calendar/SciPy packages loaded during desktop startup, and the direct dependency inventory omitted ARCH, pandas, and pandas-market-calendars. The final run above was executed after deferring those imports and reconciling the release inventory; the two focused regressions and the complete suite then passed.

25.5 2026-09-13 bounded-correction execution

After review of `df065cc`, a fresh locked Windows run covered the mean-adjusted GARCH score, the rank-zero VAR lag boundary, and independent rolling/final study comparison. Pytest collected 1,772 tests: 1,770 passed, two were skipped, and none failed or errored in 1,030.153 seconds. The JUnit artifact was `.tmp/time-series-workbench-pytest-corrections-final.xml` with SHA-256 `b41aed19d0421a5405e5a208c03d9f4c1f7cbc83e6c3143b64c186283d8267af`.

The run used Python 3.13.15 on Windows 11 (`10.0.26100`) and `uv.lock` SHA-256 `f0025bee34b23cff4834be888e4013a57d2129fc24caea13b42a7bf5a7dae1a2`. Repository-wide Ruff lint and mypy checks passed, and every changed Python file passed Ruff's format check.

CHAPTER 26

Validation: reliability and performance

Run from a clean checkout:

```
uv run python tools/run_import_benchmark.py --profile benchmarks/manifests/profiles/ci.
    json --artifact-dir .tmp/benchmark-artifacts --cache-dir .tmp/benchmark-cache --
    result .tmp/import-ci.json
uv run python tools/run_reliability_validation.py --benchmark-result .tmp/import-ci.
    json --result .tmp/reliability.json
```

The second command emits a canonical versioned `ReliabilityReport`. It runs production save publication and recovery boundaries in an isolated directory: write and publication failures must preserve the last explicit save; a spawned writer is terminated after package staging but before publication; an autosave restores; changed/truncated sources and invalid cache receipts are rejected; and a failed Desktop worker is followed by a successful retry. An explicitly supplied reliability work directory must be new or empty; the runner refuses a non-empty directory rather than overwriting unrelated files.

The report separates physical facts from deterministic injections. Existing Windows CI benchmark evidence records duration, peak RSS, cancellation, cache, and reopen metrics. `PERF-002` remains an owner-approved extrapolation from `local-4x`, not a measured 50,000-by-5,000 run; spawned cache-worker peak RSS remains residual risk. macOS/Apple Silicon is explicitly reported unavailable until the same command is run on that host.

The reference profile requires Windows 11 x64 or Apple Silicon macOS, at least 8 performance cores, 32 GB RAM, and NVMe-class storage. The accepted `local-4x` record came from Windows 11 AMD64 with 4 logical CPUs and about 10 GB RAM, below those reference preconditions; its machine-readable hardware facts remain in `benchmarks/results/import-local-4x/result.json`.

26.1 Benchmark foundation and reproducibility rules

The reproducibility foundation was established by `TBLZ-002` (recorded 2026-08-29). The versioned surface consists of deterministic dataset profiles in `benchmarks/manifests/profiles/`, JSON-emitting materialization and benchmark commands in `tools/`, and machine-readable benchmark output captured as an approved validation record when needed. Dataset generation is formula-driven and deterministic for a given seed. Directory hashes are computed from relative paths plus file bytes. Profile manifests declare the intended hardware floor for each benchmark tier, and benchmark results record hardware facts and engine versions alongside duration and peak RSS.

Use the CI profile for regression detection in automation and local smoke checks. Use the reference profile on explicitly provisioned Windows 11 x64 or Apple Silicon macOS hardware before changing published performance budgets. `benchmarks/results/` is ignored because normal benchmark artifacts can be large; move an intentionally retained result into the validation corpus with its hardware context and review.

CHAPTER 27

Validation: release evidence expansion

Engineering work based on combined `main` commit `5f45911` (industrial calibration proof plus Excel support). This increment does not merge unrelated SPC, Gage R&R or selection branches, and does not imply that their changes are covered.

27.1 Acceptance architecture

Work package	Implemented evidence/control	Boundary
Combined regression	Full-suite baseline investigation; regression coverage for any exposed defects	A focused pass does not supersede a failed or interrupted whole-suite run
Retained CI evidence	Two protocol subprocesses, raw logs, source/dependency identities, artifact checksum verification, ZIP manifest, ZIP checksum, JUnit upload	CI retention is not controlled release custody or independent approval
Interpretation correctness	Analytic OLS p/CI/effect/conclusion checks; undefined ridge inference; normal capability indices/warnings; Wilson intervals	Selected cases and methods, not every interpretation in the product
Capability workflow	Real QML/controller preparation, explicit model, assumptions, analysis, report, project persistence, stale identity and rerun	Normal continuous model, deliberately skewed fixture, HTML export
Packaged execution	Build/install wheel in a fresh Python environment, reject source-path substitution, execute both desktop workflows from installed application/QML, retain hashes, verify project after uninstall	Not a signed native installer or a clean OS image

Work package	Implemented evidence/control	Boundary
Native release acceptance	Versioned operator protocol covering two platforms, signatures, actual inputs, DPI, accessibility, export and lifecycle	Must be executed on the identified native candidates and supported clean machines

HypoNum already provides interpretations. These tests assess selected conclusions and uncertainty numerically and semantically, rather than treating interpretation presence as a new feature. They also retain those results through report adapters.

27.2 Retain a complete engineering run

```
uv sync --locked --all-groups
uv run --locked python -m pytest --junitxml=.tmp/regression.xml
uv run --locked python tools/run_validation_evidence.py --output-dir .tmp/validation-
    evidence
```

The evidence runner executes both protocols even when the first fails. It checks every declared artifact checksum before archiving. Its `manifest.json` records each child command, return code, reported status, source/dependency baseline and the observed hash of every archived file. A checksum mismatch, missing report, timeout, failed protocol or mismatched source baselines prevents an overall pass. Safe top-level diagnostic files from a failed protocol are retained as failed evidence with their observed hashes; they are not relabelled accepted outputs. Cache directories are not archived. Existing output directories cannot be reused.

The ZIP includes protocol reports, summaries, screenshots, synthetic inputs, exported reports, project packages and child process logs. Keep `evidence.zip` and `evidence.zip.sha256` together. SHA-256 detects changed bytes; it is not a digital signature, attribution record or protected audit trail.

CI runs the source regression and evidence matrix on Windows/macOS and Python 3.13/3.14. A separate Python 3.13 installed-wheel job runs on both OS runners. Uploads retain successes and failures, with platform/interpreter/commit/attempt identities. Retention is set to 90 days: transfer accepted release records to owner-controlled storage before expiry. CI configuration is not evidence that a remote job has already run or passed.

27.3 Baseline findings

The first combined-source full test run exposed an outdated plot smoke assertion and then a Windows access violation during preparation-test Qt teardown. The plot now intentionally highlights a numeric column for keyboard access but does not assign Y or run analysis. Its test checks that actual contract. Preparation integration tests now explicitly retain their QML objects and destroy windows before engines, instead of relying on Python local-variable destruction order. Neither change relaxes statistical acceptance thresholds.

The same lifecycle correction extends to bridge/report QML integration tests. Stale geometry and control-class assertions now check header/data alignment and the actual ComboBox/TextField inheritance of HypoNum's approved control subclasses, without requiring an outdated row height or an unwrapped base control.

Repository-wide formatting also has an inherited backlog in unrelated files. Keep that backlog separate from statistical acceptance and do not claim all CI gates are green merely because the new focused tests pass. Actual run outcomes, including partial/failed runs, belong in the retained execution evidence.

27.4 Native and human follow-up

Execute Chapter 13 (REL-NATIVE-001) on Windows x64 and Apple Silicon; identify the machine, binary and observer for each result. No local source/wheel test closes missing signature, notarization, clean-machine, native accessibility, release-owner custody or independent statistical review.

See Chapter 24, Chapter 23, and Chapter 12.

CHAPTER 28

Validation: module-foundation record

Date: 2026-09-14. Branch: `feature/module-foundation`. Comparison baseline: `a3ec0418bf0b89a423c3935fa0ebf2d7514ec87b` (also current main). Active worktree: `C:/Users/User7/Desktop/Tabelz/.tmp/worktrees/module-foundation` (relocated without content changes from the earlier sibling worktree). Changes are uncommitted; this is review evidence, not release acceptance.

28.1 Continuation snapshot

Before the next migration slice, the uncommitted foundation sources were captured read-only. Relevant SHA-256 values were: controller `9D18D9461ED44AD32D4A8B75CC219F4AD93F2AF9AB97A59752D4503ECA10E044`, QML bridge `78D2DC841A25895E7DF7D921FE1B4513588EDCBA6B2629A8475B23C688124D07`, main QML `E57BF0B3A13D987C9F31C6A74CCC4202661D16CBCFB900D1A97C114D18BB0B6D`, time-series engine `31872611DEDC346792F2C79CEA0C00C4C8577700E94AAB879CBB34D9218DE77C`, and time-series workbench `D5CD3B54C44AD30EFE1E544BA7DAB1462948AB33A760B9B60CD32348A00F6077`. All Python sources in that snapshot were at their documented `1.0.0` enrollment revision.

28.2 Continuation: time-series desktop lifecycle

- Added `desktop/time_series_service.py` 1.0.0 as the typed owner of request IDs, executor handoff, cancellation, loading/error/result state, and stale responses.
- Bumped `desktop/controller.py` to 1.0.1 and reduced it from 15,339 to 15,294 lines. Public attributes and slots are compatibility adapters over the service.
- Updated the base-aware version gate so an existing unversioned source may contain a later revision in the same uncommitted enrollment change only when its explicit enrollment snapshot version is recorded and both the snapshot and current history entries exist. Recorded snapshot versions become immutable once committed. New files still must start at `1.0.0`.
- Focused service/controller/QML bridge run: 48 passed in 200.90 seconds.
- Ruff lint/format and strict focused mypy passed for the new service and affected files.

28.3 Continuation: QML shell menu boundary

- Moved the unchanged File/Edit/Prepare/Explore/Analyze/Time Series/Investigate/Results/Plug-ins/Help hierarchy into `qml/ApplicationMenuBar.qml` (576 lines).
- The menu receives only the controller and active-column state and emits named navigation signals. `main.qml` keeps dialog/window navigation and the existing `ResultReportsWindow.qml` keeps results-workspace presentation ownership.
- `qml/main.qml` decreased from 14,311 to 13,879 lines; its ratchet was lowered.
- Focused runtime construction/menu interaction and QML consistency checks produced 2 passes plus the documented pre-existing `TimeSeriesWorkflow.qml:158` font-size failure. No layout or style value was changed in this slice.

28.4 Continuation: time-series preparation and validation boundaries

- Added `analysis/time_series_preparation.py` 1.0.0 (485 lines) for calendar, frequency conversion, alignment, as-of vintages, fitted transformations, and seeded back-transformation distributions.
- Added `analysis/time_series_validation.py` 1.0.0 (126 lines) for rolling-origin designs, final-evaluation isolation, sliding/expanding windows, and vintage-aware training selection.
- Bumped `analysis/time_series_workbench.py` to 1.0.2 and reduced it from 2,345 to 1,806 lines; its ratchet was lowered. Old import identities are retained, including an explicit static-type-compatible `AggregationKind` export.
- Existing workbench, controller, and architecture selection: 59 passed in 81.18 seconds. Strict focused mypy and Ruff lint/format passed for all three modules.

28.5 Continuation: application-wide report implementation provenance

- Added `provenance/implementation.py` 1.0.0 and `desktop/result_provenance.py` 1.0.0. Every current desktop report family now maps to its participating analysis implementation modules instead of treating GARCH as the only versioned result path.
- Bumped `desktop/controller.py` to 1.0.2. New report-block payloads include application version, report schema version, optional operation implementation version, and a sorted module-version map. Existing blocks are loaded byte-for-byte without relabelling.
- Moved report-only residual-lag selection out of the controller; controller size is now 15,285 lines and its ratchet was lowered again.
- Provenance/report unit, persistence, and QML selection: 19 passed in 33.43 seconds. Strict focused mypy and Ruff passed for both new provenance modules.

28.6 Continuation: manufacturing services

- Added `desktop/manufacturing_spc_service.py` 1.0.0 (102 lines) for SPC request state, stable source-row IDs, stage results/context, control-plan state, and saved manufacturing studies. The controller advanced to 1.0.3.
- Added `desktop/manufacturing_msa_service.py` 1.0.0 (111 lines) for crossed and extended gage request/result lifecycles. The controller advanced to 1.0.4.
- Added `desktop/manufacturing_capability_service.py` 1.0.0 (132 lines) for continuous/attribute validation, request identity, stale-response rejection, and result-row conversion. The controller advanced to 1.0.5.
- Added `desktop/manufacturing_specification_service.py` 1.0.0 (265 lines) for spec-file parsing and preview, mapping review, diagnostics, pending replacement, and applied specifications. The controller advanced to 1.0.6 and is now 15,179 lines; its ratchet was lowered after each net reduction.
- Focused SPC/controller/control-plan/report coverage passed 46 tests; focused MSA, gage-controller, and report coverage passed 20; specification/capability analysis, controller, persistence, bridge, and validation coverage passed 66. A combined pre-specification manufacturing/architecture/QML selection passed 102 tests.
- Strict focused mypy passed for all four services; focused Ruff lint/format passed.

28.7 Continuation: regression service

- Added `desktop/regression_service.py` 1.0.0 (355 lines) for binary, count, ordinal, nominal, and model-selection validation, background execution, state, resets, and stale-response handling. Public controller fields/methods remain explicit adapters.
- Bumped `desktop/controller.py` to 1.0.7 and reduced it to 14,940 lines; its ratchet was lowered. New reports record the participating regression service.
- Focused regression controller and full QML-bridge selection passed 40 tests; strict service mypy and focused Ruff passed.

28.8 Scope and structural checks

- 154 runtime Python files now declare their first tracked `MODULE_VERSION`.
- 144 existing modules have identical Python ASTs after removing the added version declaration. Their remaining textual changes are formatting only.
- Existing sources changed structurally during the full continuation include `desktop/controller.py`, `analysis/time_series_advanced.py`, `analysis/time_series_workbench.py`, and `qml/main.qml`; each extraction above records its compatibility boundary and focused evidence.
- The extracted `DesktopQmlBridge` class has an identical AST to its baseline class.
- The four extracted numerical fitting/testing functions have identical ASTs after normalizing only their `reproducibility` keyword expressions. Calculation code, signatures, defaults, and result fields are unchanged.
- Controller: 18,352 -> 15,339 lines. New Qt bridge: 3,037 lines (explicit legacy exception, still a migration target).
- Advanced time-series orchestration: 1,648 -> 649 lines. Four engines and shared helpers now have dedicated source files below the new-file size ceiling.
- `qml/main.qml` is now 13,879 lines after the menu-boundary extraction.
- The benchmark agents' untracked forecast-comparison code and sample-data changes in the original checkout were not included or modified.

28.9 Environment

Checks used the existing interpreter/tools at `C:/Users/User7/Desktop/Tabelz/.venv/Scripts/` with `PYTHONPATH=src` and the isolated worktree as working directory. No dependency installation or lockfile changes were made. Installed versions were compared with `uv.lock` and matched for arch, NumPy, SciPy, statsmodels, pandas, pandas-market-calendars, Polars, PySide6, pytest, Ruff, and mypy.

Below, `python`, `ruff`, and `mypy` denote those environment executables; the normal project equivalent is `uvrun--locked<command>` in an independently synced worktree.

28.10 Executed checks

Check	Result
`pythontools/check_architecture.py--basea3ec0418...`	Passed for 197 Python/QML sources; includes enrollment-snapshot/version/history, import-direction, and size gates

Check	Result
Ruff lint and format on all runtime Python plus changed tool/tests	Passed
`pythontools/generate_schemas.py--check`	Passed; no schema changes
`gitdiff--check`	Passed; Windows line-ending conversion notices only
Existing advanced-analysis, time-series controller, and QML bridge tests	62 passed in the initial focused run
New architecture/provenance tests plus updated import-order test	24 passed in the final focused run
Full desktop unit directory after continuation fixes	454 passed; two existing Qt/deprecation warnings
Final regression/provenance/startup/architecture checkpoint selection	42 passed
Full unit sweep, `python-mpytest tests/unit-oaddopts=-q--tb=short--junitxml=.tmp/module-foundation-unit.xml`	1,713 passed, 2 skipped, 1 failure in 896.42 seconds; see correction below
Broader QML/report/SQL/rename integration selection plus architecture tests	64 passed, 2 failed in 323.48 seconds; see limitations below
Strict mypy on the nine extracted/affected analysis/adapter modules with `--follow-imports=silent`	Passed
Full `python-mmypy--no-incrementalsrc/tabelz`	Six errors reproduced in the original checkout; see below

The full unit sweep's sole failure was a source-text test searching for Qt imports at their old location in `controller.py`. The test now checks engine-before-Qt bootstrap order in both the controller and the separately importable bridge. Its replacement assertions passed in isolation and in the final 24-test architecture run. The full 1,716-test unit sweep was not repeated after that test-only correction; do not report this record as a single final all-green full-unit command.

A later desktop-only sweep was interrupted at 47% for the worktree relocation. It had exposed three stale source-location assertions for the extracted menu and one eager optional-package import caused by the first provenance map. The assertions now check `ApplicationMenuBar.qml` signals plus shell handlers, and provenance imports participating modules only when a new snapshot is created. The complete rerun passed all 454 desktop unit tests; the dedicated deferred-startup test also passed.

The broader integration command selected:

```
tests/integration/test_qml_bridge.py
tests/integration/test_ui_consistency_qml.py
```

```
tests/integration/test_result_reports_qml.py
tests/integration/test_sql_qml.py
tests/integration/test_rename_columns_qml.py
tests/unit/test_architecture_contracts.py
```

28.11 Known limitations and baseline failures

1. `test_qml_uses_compact_tokens_and_native_result_window_close_controls` rejects `font.pixelSize:16` in the unchanged `qml/TimeSeriesWorkflow.qml:158`. The same assertion fails in the original checkout. This foundation does not change styling.
2. `test_successful_import_closes_preview_dialog_to_reveal_grid` returned empty `grid_rows` during the broader combined run. It passed in the initial 62-test run and in isolated reruns on both baseline and foundation. This is evidence of intermittent behavior, not proof that the underlying timing issue is fixed.
3. Full mypy reports one redundant cast in `analysis/spec_sheet.py` and five SPC typing errors in `desktop/controller.py` (three redundant casts and two optional chart accesses). All six were reproduced using `--no-incremental` on the original checkout. A post-continuation full rerun reports exactly those same six errors; the extracted modules and compatibility aliases add no remaining errors.
4. Existing numerical/Qt warnings remain, including statsmodels singular-matrix, degenerate-fit, and VECM complex-cast warnings, Qt application-class warnings, and a deprecated hover-event constructor. They were not suppressed for this refactor.

28.12 Not claimed

- The full integration/validation/release suites, installed-wheel builds, installer checks, signing, macOS checks, and the full visual/scaling matrix were not run.
- All desktop domains and the main QML shell are not yet modularized.
- Module versions are included in every newly created legacy desktop report payload; older saved report blocks remain unchanged. Non-report execution surfaces without a durable result contract still require domain-specific provenance work.
- Git branch protection is not configured by a source patch. An owner must require the architecture workflow's `module-contracts` status before treating it as a merge gate.
- No scientific correction, production-readiness declaration, or complete historical reproducibility guarantee follows from adding initial version labels.

Use the current root architecture and repository guidance for the next bounded migrations. Review and integrate this historical baseline before treating the foundation evidence as input to a new feature task.

28.13 Policy disposition for superseded modularization record

The former `docs/modularization.md` was a coordination record, not a product or compatibility contract. Its per-file `MODULE_VERSION`/history mechanism, migration queue and multi-agent ownership procedure are superseded. Authority rests with root `AGENTS.md`, current `ARCHITECTURE.md`, Git history, public schema and operation versions, implementation provenance and the no-growth architecture ratchets. This chapter preserves historical execution evidence; it makes no current release claim and does not require recreating the superseded workflow or status mechanisms.

Corpus coverage

This corpus is self-contained; the former Markdown records are historical inputs, not build-time dependencies. Coverage is organized as follows:

Source	Corpus location
Statistical method explanations and desktop SQL import	User manual
Public Python, recipes, plug-ins and internal numerical/execution contracts	This reference
Enterprise portability boundary	Enterprise prototype chapter
Desktop data and display contracts	Desktop contracts chapter
Build, signing, licensing, support, and native acceptance	Release chapters
Assets, boundaries, risks, and residual ownership	Security threat-model chapter
Development checks, plans, requirements, numerical policy, protocols, records, and release gates	Validation chapters
Typography and listings	Shared manual preamble (included once at build time)