Package 'weasel'

November 13, 2026

Type Package

Title Wave-Based Extraction and Selection for Longitudinal Data

Version 0.4.4

Description Turns wave-and-respondent selection in longitudinal panel and repeated-measures data into an explicit, reproducible workflow. Provides a scope-based pipeline for wave pattern analysis and a planning interface for scenario-based respondent selection with configurable missing-wave tolerances, interior-gap constraints, and endpoint requirements.
Wave grids may be consecutive integers or the observed schedule (e.g. biennial panels). Includes tolerance-sensitivity and attrition-selectivity diagnostics, plus a generator of ready-to-paste methods-section justification text. A full workflow guide is available at <https://siardv.github.io/weasel/>.

License MIT + file LICENSE

URL https://github.com/siardv/weasel, https://siardv.github.io/weasel/

BugReports https://github.com/siardv/weasel/issues

Encoding UTF-8

Depends R (>= 3.6)

Imports stats, tools, utils

Suggests cli, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

Author Siard van den Bosch [aut, cre, cph]

Maintainer Siard van den Bosch <siardvandenbosch@me.com>

Repository CRAN

Date/Publication 2026-11-13 16:10:02 UTC

Contents

evaluate_weasel_scope . . . 2
generate_weasel_dummy_data . . . 3
print.weasel_plan . . . 5
set_weasel_scope . . . 6
weasel_apply . . . 8
weasel_clear_scope . . . 9
weasel_compare_scenarios . . . 9
weasel_compare_to_sentence . . . 11
weasel_example . . . 12
weasel_filter_wave_summary . . . 12
weasel_get_data_by_row . . . 13
weasel_justify_subset . . . 14
weasel_logo . . . 16
weasel_match_scenario . . . 16
weasel_plan . . . 17
weasel_print_table . . . 19
weasel_reshape_to_wide . . . 20
weasel_scope_info . . . 21
weasel_selectivity . . . 22
weasel_sensitivity . . . 23
weasel_subset_to_sentence . . . 24
weasel_summarize_subset . . . 25
weasel_summarize_waves . . . 26

Index **28**

`evaluate_weasel_scope` *Evaluate wave bounds and the grid*

Description

Finalises `lower`, `upper`, and the wave grid within the active scope, and validates `min_present` against the span length. Called automatically by `weasel_reshape_to_wide()`, but can be run explicitly to inspect bounds before reshaping. With `grid = "consecutive"`, a warning (class `weasel_empty_waves`) is issued once per scope when the span contains waves that no respondent has, since such waves count as missed by everyone.

Usage

```
evaluate_weasel_scope()
```

Value

The scope environment, invisibly.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)
set_weasel_scope(d, "id", "time")
evaluate_weasel_scope()
weasel_clear_scope()
```

`generate_weasel_dummy_data` *Generate synthetic longitudinal panel data*

Description

Creates a long-format data frame in which wave-level missingness is represented the way real panel files represent it: a respondent who misses a wave simply has no row for that wave. Four participation mechanisms are layered (random skips, attention decay, attrition, and block dropout), and a small amount of item-level missingness (`NA` values inside the outcome columns of otherwise observed rows) can be added on top.

Usage

```
generate_weasel_dummy_data(
  n_ids = 1000,
  n_times = 13,
  n_vars = 5,
  prop_random = 0.05,
  prop_attention = 0.08,
  attention_center = 10,
  attention_scale = 2.5,
  prop_attrition = 0.06,
  prop_block = 0.04,
  block_duration_range = c(2, 4),
  prop_item_missing = 0.02,
  id_start = 800001,
  waves = NULL,
  seed = NULL
)
```

Arguments

`n_ids`	Number of respondents; an integer from 1 to 2147483647.
`n_times`	Number of time points (waves); an integer from 0 to 2147483647, validated before `waves` replaces its value. When `waves` is not supplied, `n_times` must be greater than 2.
`n_vars`	Number of outcome variables to generate; an integer from 1 to 2147483647.

prop_random
: Probability that any given (respondent, wave) observation is skipped at random. Like every prop_* parameter, it must be a single number in [0, 1]; invalid values fail immediately, before any random draw.

prop_attention
: Asymptotic probability of attention-related wave skipping at late waves.

attention_center
: Wave position at which the attention-decay curve is centred. Must be a single finite number.

attention_scale
: Steepness of the attention-decay curve; larger values give a more gradual increase. Must be a single finite number greater than zero.

prop_attrition
: Proportion of respondents who permanently drop out from a random wave onwards.

prop_block
: Proportion of respondents who miss one contiguous block of waves.

block_duration_range
: Integer vector of length 2 giving the min/max duration of block missingness. Equal endpoints request a fixed duration. Blocks extending beyond the final scheduled wave are clipped at that wave.

prop_item_missing
: Probability that an individual outcome value on an observed row is NA (item nonresponse).

id_start
: Starting integer for respondent identifiers; must be between -2147483647 and 2147483647. The final identifier, id_start + n_ids - 1, must not exceed 2147483647. Invalid ID ranges are rejected before random draws.

waves
: Optional integer vector of wave labels, for example seq(2008, 2032, by = 2) for a biennial schedule. Values must be finite and between -2147483647 and 2147483647. Deviations of up to 1e-8 from an integer are accepted and rounded; the sorted, unique schedule must contain more than 2 labels. When supplied it overrides n_times; the participation mechanisms operate on the positions of this schedule, and the returned time column contains these labels. Analyse such data with grid = "observed" in weasel_plan() or set_weasel_scope().

seed
: Random seed; a single integer-valued number between -2147483647 and 2147483647 (fractional and non-numeric values are rejected before set.seed() is reached). If NULL, a seed is drawn in the separate process and reported so the data set can be regenerated. Repeated calls with seed = NULL can produce different data. A given explicit seed reproduces the same panel regardless of the caller's RNGkind(); see Details.

Details

The returned grid is therefore deliberately incomplete; this is what both weasel pipelines analyse. Synthetic draws run in a separate R process from the same R installation. The caller's .Random.seed, RNGkind() configuration and cached Box-Muller normal deviate remain untouched; no bindings are created or changed in the caller's global environment. The generator pins the separate process to RNGkind("Mersenne-Twister", "Inversion", sample.kind = "Rejection"), the R >= 3.6 defaults, so an explicit seed reproduces the same panel regardless of the caller's RNG configuration. Each call incurs R process startup time. Temporary request, script and result files are removed on exit, including when generation fails.

Value

A data frame in long format with columns `id`, `time`, and `var1` ... `varN`. Respondents keep at least one observed wave, so the number of distinct ids always equals `n_ids`, but `nrow(result)` is typically smaller than `n_ids * n_times`.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)

# the grid is incomplete: wave-level missingness is row absence
nrow(d) < 50 * 8

# scope pipeline starts here
set_weasel_scope(d, "id", "time")
weasel_clear_scope()

# plan pipeline starts here
p <- weasel_plan(d, "id", "time", span = "core")

# a biennial schedule; analyse with grid = "observed"
b <- generate_weasel_dummy_data(n_ids = 40, waves = seq(2008, 2020, 2),
                                seed = 1)
pb <- weasel_plan(b, "id", "time", span = "full", grid = "observed")
```

`print.weasel_plan` *Print a weasel plan*

Description

Compact display of a plan object: the chosen span, the number of respondents observed in it, and the scored scenario table. The attached data (if any) and the per-scenario id lists are summarised by size instead of being printed in full.

Usage

```
## S3 method for class 'weasel_plan'
print(x, digits = 3, ...)
```

Arguments

`x`	Object returned by `weasel_plan()`.
`digits`	Integer; decimal places for numeric columns.
`...`	Ignored.

Value

`x`, invisibly.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 60, n_times = 8, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
p
```

`set_weasel_scope` *Set up a weasel scope for wave-pattern analysis*

Description

Initialises the package-internal scope that holds data, column names, and wave-range parameters used by the downstream pipeline functions. All arguments are validated immediately, so mistakes fail here rather than several steps later. The user's global environment is never modified.

Usage

```
set_weasel_scope(
  data,
  id,
  wave,
  min_present = 1L,
  max_missing = NULL,
  max_gap_len = NULL,
  n_gap_max = NULL,
  require_endpoints = FALSE,
  lower = NULL,
  upper = NULL,
  grid = c("consecutive", "observed"),
  override = TRUE,
  size = NULL,
  gap = NULL,
  n_gap = NULL
)
```

Arguments

`data`	A data frame in long format. A respondent is considered observed at a wave if a row with that (id, wave) pair exists.
`id`	Name of the respondent-identifier column. Any atomic type (integer, character, ...) is supported.
`wave`	Name of the wave/time column. Must be numeric with integer-valued entries. Non-missing values must be finite and between -2147483647 and 2147483647. Deviations of up to `1e-8` from an integer are accepted and rounded for participation and duplicate checks.

`min_present` Single integer from 1 to 2147483647 (default 1); respondents with fewer observed waves inside the span are dropped by `weasel_reshape_to_wide()`. The default keeps every respondent with at least one observed wave, so exploration shows the full participation landscape unless you narrow it.

`max_missing` Optional integer from 0 to 2147483647; maximum allowed number of missing waves inside the span. `NULL` (default) applies no constraint.

`max_gap_len` Optional integer from 0 to 2147483647; maximum allowed length of an interior gap (a run of missing waves strictly between a respondent's first and last observed wave inside the span). `NULL` (default) applies no constraint.

`n_gap_max` Optional integer from 0 to 2147483647; maximum allowed number of interior gaps. `NULL` (default) applies no constraint.

`require_endpoints` If `TRUE`, only respondents observed at both the first and last wave of the span are kept. Default `FALSE`.

`lower` Optional lower bound of the wave range; an integer between -2147483647 and 2147483647.

`upper` Optional upper bound of the wave range; an integer between -2147483647 and 2147483647.

`grid` How the wave grid inside the span is defined. `"consecutive"` (default) treats every integer between `lower` and `upper` as a scheduled wave; `"observed"` uses only wave values that occur anywhere in the data, which is the right choice for biennial and other non-consecutive schedules. With `"consecutive"`, `evaluate_weasel_scope()` warns when the span contains waves that no respondent has.

`override` If `TRUE`, overwrite any existing scope. If `FALSE` and a scope already exists, an error is raised.

`size`, `gap`, `n_gap` Deprecated aliases (classed warning `weasel_deprecated`): use `min_present` (for `size`, whose minimum is used), `max_gap_len` (for `gap`), and `n_gap_max` (for `n_gap`). An explicitly supplied new-name argument takes precedence over its alias. When used, `size` must be a nonempty numeric vector of finite values at least 1 and within `1e-8` of integers. Its rounded minimum must be at most 2147483647; larger nonminimum values are allowed.

Value

The scope environment, invisibly.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)

# step 1 of the scope pipeline: bind data and define the wave range
set_weasel_scope(d, "id", "time", upper = 8)

# only respondents whose interior gaps are at most 1 wave long
set_weasel_scope(d, "id", "time", max_gap_len = 1)

# biennial schedule: use the observed grid
```

```
b <- generate_weasel_dummy_data(n_ids = 30, waves = seq(2008, 2018, 2),
                                seed = 1)
set_weasel_scope(b, "id", "time", grid = "observed")

weasel_clear_scope()
```

`weasel_apply` *Apply a scenario to obtain filtered long-format data*

Description

Subsets the original data to the respondents and wave range selected by a given scenario. This is typically the final step: after reviewing `weasel_compare_scenarios()`, pick a scenario and call this to get the analysis-ready data frame. Selection metrics count each (id, wave) pair once, but output rows are returned as they appear in the data; if duplicated pairs remain in the result, a classed warning (`weasel_duplicates`) reminds you to resolve them before modelling.

Usage

```
weasel_apply(plan_obj, scenario, data = NULL)
```

Arguments

`plan_obj`	Object returned by `weasel_plan()`.
`scenario`	Name (or unambiguous abbreviation) of the scenario.
`data`	Optional long-format data frame; defaults to the data attached to `plan_obj`. Required when the plan was created with `keep_data = FALSE`. Explicitly supplied data are compared with the plan's structural fingerprint; a mismatch triggers a classed warning (`weasel_data_mismatch`), since the plan's id lists were computed on the original data.

Value

A data frame.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")

balanced <- weasel_apply(p, "anchored_balanced")
dim(balanced)

# unambiguous prefixes work
strict <- weasel_apply(p, "anchored_s")
dim(strict)
```

```
# plans built with keep_data = FALSE need the data passed back in
p2 <- weasel_plan(d, "id", "time", span = "core", keep_data = FALSE)
nrow(weasel_apply(p2, "lenient", data = d))
```

`weasel_clear_scope` *Clear the active weasel scope*

Description

Removes the scope created by `set_weasel_scope()`. Useful at the end of scripts, examples, and tests.

Usage

```
weasel_clear_scope()
```

Value

`TRUE` invisibly if a scope was removed, `FALSE` otherwise.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 20, n_times = 5, seed = 1)
set_weasel_scope(d, "id", "time")
weasel_clear_scope()
```

`weasel_compare_scenarios`
Score and rank scenarios from a weasel plan

Description

Computes a composite score for each scenario from *observed* subset quality, never from the scenario's configured tolerances. With the default weights the score is

Usage

```
weasel_compare_scenarios(plan_obj, weights = NULL, tie_tolerance = 0.01)
```

Arguments

plan_obj	A list with element plan (data frame of scenarios), as returned by weasel_plan().
weights	Named numeric vector overriding any of the default weights c(coverage = 2, endpoints = 1.2, size = 0.8, missing = 0.6, gaps = 0.4). NA values are rejected. Zero and negative weights are accepted deliberately: a zero weight removes a term from the score, and a negative weight inverts the direction of its contribution.
tie_tolerance	Single non-negative number; scenarios whose scores are within this distance of the best score are flagged as near_tie (only when at least two qualify). Must be supplied as an actual numeric scalar: vectors, NA, and character values are rejected rather than silently reduced or converted.

Details

$$2 \cdot coverage + 1.2 \cdot endpoints + 0.8 \cdot size - 0.6 \cdot missing - 0.4 \cdot gaps$$

where coverage is mean_prop_present, endpoints is endpoint_rate, size is n_ids divided by the largest n_ids across scenarios, missing is worst_missing / L (the largest per-respondent count of missing waves among retained respondents, normalised by span length), and gaps is (mean_n_gap + mean_max_gap) / L. Scenarios retaining no respondents receive NA and are never recommended. Exact score ties are broken in favour of the larger sample, then by input-table order; tie_tolerance only flags near-ties in near_tie and never changes which scenario is recommended.

The score is a configurable heuristic, not a validated decision rule: the weights encode one reasonable trade-off, and the recommended flag marks the highest score under the declared weights, nothing more. Because the size term is normalised by the largest n_ids in the supplied set, a scenario's score depends on which scenarios it is compared with; scores are comparison-relative, not absolute properties of a scenario. With subsets and weights held fixed, a change to the comparison set affects existing scores only through the size contribution, so existing scores change only when the largest n_ids in the set changes, and such a change can reverse the ranking, and hence the recommendation, between two scenarios whose own subsets are unchanged. When the largest n_ids is unchanged, existing scores are unchanged, although an added scenario can itself become the recommended one. A zero size weight removes this dependence; scenarios retaining no respondents keep NA in every comparison. Scenarios whose scores fall within tie_tolerance of the best are flagged in near_tie, signalling that the heuristic does not meaningfully distinguish them; endpoint-requiring scenarios earn the endpoints term by construction, which is worth keeping in mind when comparing anchored with unanchored scenarios.

Value

The plan data frame with added score, recommended, and near_tie columns. The active weights are attached as attribute "weights" and the per-scenario score decomposition as attribute "score_components", so every recommendation can be audited.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
```

```
p <- weasel_plan(d, "id", "time", span = "core")
cmp <- weasel_compare_scenarios(p)
weasel_print_table(cmp, title = "Scenario comparison")

# emphasise sample size more strongly
cmp2 <- weasel_compare_scenarios(p, weights = c(size = 2))
cmp2$scenario[cmp2$recommended]

# audit the recommendation: active weights and score decomposition
attr(cmp, "weights")
attr(cmp, "score_components")
```

`weasel_compare_to_sentence` *Summarize scenario comparison as a sentence*

Description

Produces a human-readable sentence from the scored scenario table, suitable for reports or console output. Intended to be called right after `weasel_compare_scenarios()`. When no scenario is recommended (for example because every scenario retains zero respondents), the sentence says so instead of naming one.

Usage

```
weasel_compare_to_sentence(cmp, digits = 3)
```

Arguments

`cmp`	Data frame returned by `weasel_compare_scenarios()`.
`digits`	Number of decimal places.

Value

A single character string.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
cmp <- weasel_compare_scenarios(p)
cat(weasel_compare_to_sentence(cmp), "\n")
```

weasel_example *Run a full weasel demonstration*

Description

Exercises both the scope-based pipeline and the planning interface on synthetic dummy data, including the sensitivity diagnostic, then cleans up the scope. Display is suppressible with `suppressMessages()` or `options(weasel.verbose = FALSE)`; results are still returned.

Usage

```
weasel_example(seed = 42)
```

Arguments

seed	Random seed passed to `generate_weasel_dummy_data()`.

Value

A list with elements `data`, `plan`, `compare`, `summary`, and `sensitivity`, invisibly.

Examples

```
res <- weasel_example(seed = 42)
```

weasel_filter_wave_summary *Filter the wave-pattern summary*

Description

Narrows the pattern table produced by `weasel_summarize_waves()`. The `pattern` column keeps its original values, so pattern ids identified in a filtered table can be passed directly to `weasel_get_data_by_row()`.

Usage

```
weasel_filter_wave_summary(n_range = NULL, ids_range = NULL)
```

Arguments

n_range	Optional length-2 numeric vector giving the min/max number of observed waves per pattern (the `n` column).
ids_range	Optional length-2 numeric vector giving the min/max respondent count per pattern (the `ids` column).

Value

A filtered data frame.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)
set_weasel_scope(d, "id", "time")
evaluate_weasel_scope()
weasel_reshape_to_wide()
weasel_summarize_waves()

# only patterns with at least 6 observed waves
weasel_print_table(weasel_filter_wave_summary(n_range = c(6, Inf)))

# only patterns shared by at least 3 respondents
weasel_print_table(weasel_filter_wave_summary(ids_range = c(3, Inf)))
weasel_clear_scope()
```

---

weasel_get_data_by_row

Retrieve long-format data matching wave-pattern rows

Description

After `weasel_summarize_waves()` has grouped respondents into patterns, this function extracts the long-format rows for every respondent who matches the given pattern row(s). Use `weasel_filter_wave_summary()` first to identify which patterns to request; the `pattern` ids it shows remain valid selectors here.

Usage

```
weasel_get_data_by_row(i = 1L, within_span = FALSE)
```

Arguments

`i`	Pattern selector: an integer vector of pattern ids (the `pattern` column of the summary, which equals the row numbers of the full, unfiltered table), or a character vector of pattern strings (`waves` values). Respondents matching any selected pattern are returned. Defaults to the first pattern.
`within_span`	If `TRUE`, only rows whose wave lies inside the scoped span are returned. If `FALSE` (default), all rows of the matching respondents are returned, including waves outside the span.

Value

A data frame (subset of the original long data).

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)
set_weasel_scope(d, "id", "time")
evaluate_weasel_scope()
weasel_reshape_to_wide()
weasel_summarize_waves()

subset1 <- weasel_get_data_by_row(1)
weasel_print_table(head(subset1), title = "Row-1 data preview")

# respondents matching either of the two most common patterns
subset12 <- weasel_get_data_by_row(c(1, 2))
weasel_clear_scope()
```

---

weasel_justify_subset *Generate a justification paragraph for a chosen scenario*

Description

Produces a plain-text paragraph that researchers can paste into a methods section to explain and justify their subset selection. The text references the structural constraints (endpoints, interior gaps, span length), frames the choice as a coverage-vs-sample-size trade-off, and, unless disabled, appends an in-text citation whose author and year are derived automatically from the package's own citation metadata (`utils::citation()`). A scenario that retains no respondents cannot be justified and raises a classed error (`weasel_error_empty_scenario`).

Usage

```
weasel_justify_subset(
  plan_obj,
  scenario = "anchored_balanced",
  style = c("methods", "concise", "extended"),
  cite = TRUE,
  author = NULL,
  year = NULL,
  package = "weasel",
  acronym = "WEASEL",
  full_name = "Wave-based Extraction and Selection for Longitudinal Data",
  digits = 3
)
```

Arguments

`plan_obj`	Object returned by `weasel_plan()`.
`scenario`	Name (or unambiguous abbreviation) of the scenario.

style	One of "methods" (full methods-section paragraph), "concise" (short summary), or "extended" (detailed rationale).
cite	If TRUE, append an in-text citation. The author and year are filled in automatically from the package's citation metadata; a plain (R package <name>) reference is used only when that metadata is unavailable.
author	Optional author string overriding the automatically derived citation author.
year	Optional year string overriding the automatically derived citation year.
package	Package whose citation metadata is used, and the name shown in the fallback reference.
acronym	Acronym for the framework.
full_name	Full name of the framework.
digits	Number of decimal places for numeric values.

Details

All styles describe the recorded window-selection method: user-supplied bounds for explicit windows, the full wave grid for full spans, or the highest total count of distinct respondent-wave pairs among equal-length contiguous grid windows for core spans (earliest window on ties). These descriptions do not establish when bounds were chosen, sample stability, or methodological superiority. If the plan does not record a selection method, that description is omitted.

A tolerance of Inf is described as no separate limit on that dimension. Other criteria (including endpoint requirements and finite tolerances) still apply. Missing tolerance metadata is not treated as unlimited.

All styles report the recorded planning population: respondents observed at least once within the analysis window, out of all distinct respondents in the supplied data. Retention is relative to the in-window population. This description uses stored plan metadata and does not require attached data. Older plans without population metadata omit the description.

Value

A single character string ready to paste into a manuscript.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 200, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")

# the in-text citation (package author and year) is automatic
cat(weasel_justify_subset(p, "anchored_balanced"), "\n")
cat(weasel_justify_subset(p, "anchored_strict", style = "concise"), "\n")
cat(weasel_justify_subset(p, "anchored_balanced", style = "extended"), "\n")

# override the derived citation, or omit it entirely
cat(weasel_justify_subset(p, "anchored_balanced",
                          author = "Custom Name", year = "2030"), "\n")
cat(weasel_justify_subset(p, "anchored_balanced", cite = FALSE), "\n")
```

weasel_logo *Display the weasel ASCII logo*

Description

Displays a coloured ASCII art logo as a message. Colours rotate without using R's random-number generator. Suppress the display with `suppressMessages()` or `options(weasel.verbose = FALSE)`.

Usage

```
weasel_logo()
```

Value

`NULL`, invisibly.

Examples

```
weasel_logo()
```

weasel_match_scenario *Match a scenario name (possibly abbreviated) to available choices*

Description

Used internally by `weasel_apply()` and `weasel_summarize_subset()`, but also useful when building custom workflows. Accepts either an exact name or an unambiguous prefix; arbitrary substrings are not matched, so an abbreviation always selects the scenario it visibly starts.

Usage

```
weasel_match_scenario(scenario, choices)
```

Arguments

`scenario`	Character string to look up.
`choices`	Character vector of valid scenario names.

Value

The matched scenario name.

Examples

```
choices <- c("anchored_strict", "anchored_balanced", "lenient_info_max")

weasel_match_scenario("anchored_strict", choices)
weasel_match_scenario("lenient", choices)  # unique prefix

# ambiguous prefix errors: "anchored" starts two scenarios
try(weasel_match_scenario("anchored", choices))
```

`weasel_plan` *Build a scenario-based selection plan*

Description

Evaluates three default scenarios (`anchored_strict`, `anchored_balanced`, `lenient_info_max`) or a validated custom scenario table against the data, computing per-scenario respondent counts and observed quality metrics. Gap constraints refer to interior gaps: runs of missing grid waves strictly between a respondent's first and last observed wave inside the span; missing endpoints are handled separately through `require_endpoints`.

Usage

```
weasel_plan(
  data,
  id,
  wave,
  span = c("core", "full"),
  core_len = 6L,
  scenarios = NULL,
  grid = c("consecutive", "observed"),
  keep_data = TRUE,
  lower = NULL,
  upper = NULL
)
```

Arguments

`data`	A long-format data frame. A respondent is considered observed at a wave if a row with that (id, wave) pair exists. Duplicated (id, wave) rows are counted once and trigger a warning. The check covers non-missing pairs in all supplied rows, including rows outside the selected window.
`id`	Name of the respondent-identifier column. Any atomic type is supported.
`wave`	Name of the wave/time column. Must be numeric with integer-valued entries. Non-missing values must be finite and between -2147483647 and 2147483647. Deviations of up to `1e-8` from an integer are accepted and rounded for participation and duplicate checks.

span — Either "core" (window of core_len consecutive grid waves with the highest total respondent-wave coverage; exact coverage ties are resolved in favour of the earliest window with a classed warning, weasel_tied_windows, and every candidate window is stored in the returned span_candidates table) or "full" (the full wave grid). Ignored when explicit lower/upper bounds are supplied; supplying both raises an error.

core_len — Desired window length when span = "core"; NULL uses the default of 6. Other values must be single integers from 0 to 2147483647 before clamping to the feasible range (2 through the number of grid waves). Any adjustment is reported in a verbose-mode message.

scenarios — Optional data frame of custom scenarios with the columns scenario, require_endpoints, max_missing, n_gap_max, max_gap_len. The table is validated; missing columns raise an error. The pre-0.4 column name max_gap_max is still accepted with a deprecation warning (class weasel_deprecated). Tolerance columns may arrive as numbers, numeric-looking character values, or factors; factors are converted through their displayed labels, never through their internal level codes, so a validated table always carries exactly the tolerances the user wrote. Uninterpretable, negative, or fractional values are rejected with an error naming the column and the offending value (Inf still means "no constraint"). require_endpoints accepts logical values, the imported labels "TRUE"/"FALSE"/"T"/"F", or the exact numbers 0/1; broader coercions such as 2 or "yes" are rejected.

grid — How the wave grid inside the span is defined. "consecutive" (default) treats every integer between the span bounds as a scheduled wave; "observed" uses only wave values that occur anywhere in the data, which is the right choice for biennial and other non-consecutive schedules. With "consecutive", a warning (class weasel_empty_waves) is issued when the chosen span contains waves that no respondent has, since such waves count as missed by everyone.

keep_data — If TRUE (default), the original data is attached to the returned object so that weasel_apply(), weasel_summarize_subset(), and weasel_selectivity() can reuse it. Set to FALSE to keep the plan object small (for example before saving it with saveRDS()); those functions then need the data passed back in through their data argument.

lower, upper — Optional explicit integer window bounds between -2147483647 and 2147483647. When either is supplied the plan records span_reason = "explicit". The justification text describes the bounds as user-supplied, without inferring when or why they were chosen. Bounds are interpreted on the chosen grid; the effective bounds are the first and last grid waves inside the requested range.

Value

A list of class weasel_plan with elements plan (scored scenario table), id_metrics, lower, upper, span (integer vector of grid waves), grid, span_reason ("core", "full", or "explicit"), span_candidates (for core spans, every candidate window with its coverage and a chosen flag; NULL otherwise), population (the planning denominator: rows and distinct ids in the data, ids and unique pairs observed in the span; all retention proportions are relative to "observed_in_span"), fingerprint (a structural fingerprint of the data: aggregate counts plus an order-invariant digest of the deduplicated (id, wave) assignments across the full supplied panel, excluding covariate values;

new fingerprints record `encoding_version = 2` and preserve supported ID values exactly, independently of numeric display options. Saved fingerprints without a version retain their historical encoding and limitations; rebuild from verified original data to create a new fingerprint. A mismatch warns without replanning or changing stored scenario IDs; the guard applies only when data are supplied explicitly to `weasel_apply()`, `weasel_summarize_subset()`, or `weasel_selectivity()`, since the attached-data path cannot mismatch by construction), `id`, and `wave`. When `keep_data = TRUE` the original `data` is attached as an attribute. Printing the object shows a compact summary instead of the raw list.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
p <- weasel_plan(d, id = "id", wave = "time", span = "core")
p

cmp <- weasel_compare_scenarios(p)
weasel_print_table(cmp, title = "Scenario overview")

sub <- weasel_apply(p, "anchored_balanced")
dim(sub)

# biennial schedule: evaluate presence on the observed grid
b <- generate_weasel_dummy_data(n_ids = 60, waves = seq(2008, 2020, 2),
                                seed = 1)
pb <- weasel_plan(b, "id", "time", span = "full", grid = "observed")

# a user-supplied window: specify the bounds directly
pe <- weasel_plan(d, "id", "time", lower = 3, upper = 8)
pe$span_reason
```

`weasel_print_table` *Print a data frame as a formatted table*

Description

The standard display function used at every stage of both pipelines. Output is produced with base R only, so it is identical regardless of which optional packages are installed. List columns (such as the per-scenario `ids` column of a plan table) are hidden from display and reported by name instead. Set `options(weasel.verbose = FALSE)` to suppress display while retaining the formatted return value.

Usage

```
weasel_print_table(x, title = NULL, digits = 3, n = NULL)
```

Arguments

x	A data frame to print.
title	Optional character string displayed as a heading above the table.
digits	Integer from 0 to 2147483647; number of decimal places for numeric columns.
n	Optional integer from 0 to 2147483647; maximum number of rows to show. NULL shows all rows.

Value

Invisibly returns the displayed (possibly truncated and rounded) data frame, without hidden list columns.

Examples

```
df <- data.frame(a = 1:5, b = rnorm(5))
weasel_print_table(df, title = "Example", digits = 2, n = 3)

# typical use: display scenario comparison from the plan pipeline
d <- generate_weasel_dummy_data(n_ids = 60, n_times = 8, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
weasel_print_table(weasel_compare_scenarios(p), title = "Scenarios")
```

---

weasel_reshape_to_wide

Reshape scoped data to a wide presence matrix

Description

Builds a respondent x wave data frame where each cell contains the wave number if the respondent is observed, and NA otherwise. Respondents with fewer than min_present observed waves are dropped, and the optional max_missing, max_gap_len, n_gap_max, and require_endpoints constraints from set_weasel_scope() are applied. Duplicated (id, wave) rows are counted once and trigger a warning.

Usage

```
weasel_reshape_to_wide()
```

Value

The pivot data frame, invisibly.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)
set_weasel_scope(d, "id", "time")
evaluate_weasel_scope()
weasel_reshape_to_wide()
weasel_clear_scope()
```

`weasel_scope_info` *Inspect the active weasel scope*

Description

Displays a compact status report of the active scope: the bound data, the grid and span, the structural constraints, and how far the pipeline has progressed. Reports "no active weasel scope." when no scope is set. Suppress the message with `suppressMessages()` or `options(weasel.verbose = FALSE)`.

Usage

```
weasel_scope_info()
```

Value

Invisibly, a list with the scope settings (`id`, `wave`, `grid`, `lower`, `upper`, `span`, `min_present`, `max_missing`, `max_gap_len`, `n_gap_max`, `require_endpoints`) and the progress counters (`n_rows`, `n_kept`, `n_patterns`), or `NULL` when no scope is active.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 30, n_times = 6, seed = 1)
set_weasel_scope(d, "id", "time", max_gap_len = 1)
weasel_scope_info()
weasel_reshape_to_wide()
weasel_summarize_waves()
weasel_scope_info()
weasel_clear_scope()
```

`weasel_selectivity` *Compare retained and excluded respondents on covariates*

Description

Selecting respondents by participation completeness can bias a sample when retention is related to substantive characteristics. This diagnostic compares the respondents a scenario retains with those who are observed in the span but excluded, covariate by covariate, and reports the standardized mean difference (SMD) for each.

Usage

```
weasel_selectivity(
  plan_obj,
  scenario,
  vars = NULL,
  data = NULL,
  at = c("first", "mean")
)
```

Arguments

`plan_obj`	Object returned by `weasel_plan()`.
`scenario`	Name (or unambiguous abbreviation) of the scenario.
`vars`	Character vector of covariate columns to compare. Defaults to every numeric or logical column except the id and wave columns.
`data`	Optional long-format data frame; defaults to the data attached to `plan_obj`. Required when the plan was created with `keep_data = FALSE`. Explicitly supplied data are checked against the plan's structural fingerprint (classed warning `weasel_data_mismatch` on mismatch).
`at`	Either `"first"` (value at the first observed wave in the span) or `"mean"` (mean over observed waves in the span).

Details

For each respondent, `at = "first"` uses the covariate value at their first observed wave inside the span. A missing item at that wave remains missing; the diagnostic does not search later waves for a non-missing value. First observed waves may differ between respondents, so this need not compare a common baseline.

With `at = "mean"`, the respondent summary averages non-missing covariate values over their observed waves in the span. Respondents can contribute different sets of waves. If a covariate changes over time, differing observation times can contribute to differences under either summary.

Duplicated (id, wave) rows trigger a classed warning (`weasel_duplicates`). Non-missing covariate values are averaged within each pair first, so duplicate physical rows do not add wave weight.

Respondents with missing summaries are omitted separately for each covariate; the reported group counts therefore need not equal the scenario's total respondent counts.

The SMD divides the retained-minus-excluded mean difference by `sqrt((sd_retained^2 + sd_excluded^2) / 2)`, using the two group sample standard deviations. It is `NA` when either group has fewer than two non-missing respondent summaries, or when this denominator is zero or `NA`. One constant group can yield a finite SMD when the other group varies.

Absolute SMD values around 0.1 are a descriptive balance heuristic whose interpretation depends on context and covariate importance. The diagnostic describes observed mean imbalance; small SMDs do not establish similar distributions, an attrition mechanism, ignorable attrition or unbiased inference.

Value

A data frame with one row per covariate and the columns `variable`, `n_retained`, `n_excluded` (respondents with a non-`NA` value), `mean_retained`, `mean_excluded`, `diff` (retained minus excluded), and `smd`, sorted by absolute SMD, largest first.

References

Austin, P. C. (2009). Balance diagnostics for comparing the distribution of baseline covariates between treatment groups in propensity-score matched samples. doi:10.1002/sim.3697.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 200, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
sel <- weasel_selectivity(p, "anchored_balanced")
weasel_print_table(sel, title = "Retained vs excluded", digits = 3)
```

`weasel_sensitivity` *Sweep selection tolerances and count retained respondents*

Description

Evaluates every combination of the supplied tolerance values against the per-respondent metrics stored in a plan, reporting how many respondents each combination would retain. Because the sweep reuses the metrics that `weasel_plan()` already computed, it is fast even for large panels and works when the plan was created with `keep_data = FALSE`. Use it to show how sensitive the sample size is to the chosen tolerances before committing to a scenario.

Usage

```
weasel_sensitivity(
  plan_obj,
  require_endpoints = c(TRUE, FALSE),
  max_missing = 0:3,
```

```
  n_gap_max = 0:2,
  max_gap_len = 0:2,
  max_gap_max = NULL
)
```

Arguments

plan_obj	Object returned by weasel_plan().
require_endpoints	Logical values to evaluate. Must be actual logical TRUE/FALSE values; numeric or character stand-ins (for example 2 or "TRUE") are rejected rather than coerced.
max_missing	Integer tolerances for missing waves in the span.
n_gap_max	Integer tolerances for the number of interior gaps.
max_gap_len	Integer tolerances for the longest interior gap.
max_gap_max	Deprecated alias for max_gap_len (classed warning weasel_deprecated); ignored when max_gap_len is supplied explicitly.

Details

Tolerance vectors must be nonempty and numeric, with finite, non-negative values within 1e-8 of integers from 0 to 2147483647. Accepted values are rounded, sorted and deduplicated. Inf is rejected here; scenario tables supplied to weasel_plan() separately support unlimited tolerances.

Value

A data frame with one row per combination and the columns require_endpoints, max_missing, n_gap_max, max_gap_len, n_ids, prop_ids (share of all respondents observed in the span), and mean_prop_present (coverage among the retained respondents; NA when nobody is retained).

Examples

```
d <- generate_weasel_dummy_data(n_ids = 150, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
sens <- weasel_sensitivity(p, max_missing = 0:2)
weasel_print_table(sens, title = "Tolerance sweep", n = 12)
```

---

weasel_subset_to_sentence

Extract the sentence from a subset summary

Description

Convenience accessor for the natural-language summary built by weasel_summarize_subset().

Usage

```
weasel_subset_to_sentence(s)
```

Arguments

s	Object returned by `weasel_summarize_subset()`.

Value

A character string.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")
s <- weasel_summarize_subset(p, "anchored_strict")
cat(weasel_subset_to_sentence(s), "\n")
```

`weasel_summarize_subset` *Summarize a chosen scenario subset*

Description

Computes headline statistics, per-wave coverage, and a missingness distribution for a specific scenario. Use this to audit a scenario before committing to `weasel_apply()`. By default the data and column names stored in the plan object are reused; supply `data`, `id`, or `wave` only to override them. `headline$n_rows` counts physical rows; when duplicated (id, wave) rows are present it therefore exceeds the deduplicated pair count, and a classed warning (`weasel_duplicates`) is emitted.

Usage

```
weasel_summarize_subset(
  plan_obj,
  scenario,
  data = NULL,
  id = NULL,
  wave = NULL,
  digits = 3
)
```

Arguments

plan_obj	Object returned by weasel_plan().
scenario	Name of the scenario to summarize.
data	Optional long-format data frame; defaults to the data attached to plan_obj. Required when the plan was created with keep_data = FALSE. Explicitly supplied data are checked against the plan's structural fingerprint (classed warning weasel_data_mismatch on mismatch).
id	Optional id column name; defaults to plan_obj$id.
wave	Optional wave column name; defaults to plan_obj$wave.
digits	Number of decimal places for the sentence output.

Value

A list with elements headline, per_wave_coverage, missing_distribution, data, and sentence.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 100, n_times = 10, seed = 1)
p <- weasel_plan(d, "id", "time", span = "core")

s <- weasel_summarize_subset(p, "anchored_balanced")
weasel_print_table(s$headline, title = "Headline")
weasel_print_table(s$per_wave_coverage, title = "Per-wave coverage")
cat(weasel_subset_to_sentence(s), "\n")
```

---

weasel_summarize_waves

Summarize wave patterns in the scoped data

Description

Groups respondents by their observed-wave pattern and counts how many share each pattern. Requires weasel_reshape_to_wide() to have been called. In the result, pattern is a stable pattern id (the row number of the full summary; filtering with weasel_filter_wave_summary() keeps it, so it can always be passed to weasel_get_data_by_row()), waves is the pattern (a . marks a missing wave), n is the number of observed waves in that pattern, and ids is the number of respondents sharing it. Rows are sorted by ids, then n, both descending.

Usage

```
weasel_summarize_waves()
```

Value

The summary data frame, invisibly.

Examples

```
d <- generate_weasel_dummy_data(n_ids = 50, n_times = 8, seed = 1)
set_weasel_scope(d, "id", "time")
evaluate_weasel_scope()
weasel_reshape_to_wide()
weasel_summarize_waves()
weasel_clear_scope()
```

Index

evaluate_weasel_scope, 2
evaluate_weasel_scope(), *7*

generate_weasel_dummy_data, 3
generate_weasel_dummy_data(), *12*

print.weasel_plan, 5

set_weasel_scope, 6
set_weasel_scope(), *4*, *9*, *20*

weasel_apply, 8
weasel_apply(), *16*, *18, 19*, *25*
weasel_clear_scope, 9
weasel_compare_scenarios, 9
weasel_compare_scenarios(), *8*, *11*
weasel_compare_to_sentence, 11
weasel_example, 12
weasel_filter_wave_summary, 12
weasel_filter_wave_summary(), *13*, *26*
weasel_get_data_by_row, 13
weasel_get_data_by_row(), *12*, *26*
weasel_justify_subset, 14
weasel_logo, 16
weasel_match_scenario, 16
weasel_plan, 17
weasel_plan(), *4, 5*, *8*, *10*, *14*, *22–24*, *26*
weasel_print_table, 19
weasel_reshape_to_wide, 20
weasel_reshape_to_wide(), *2*, *7*, *26*
weasel_scope_info, 21
weasel_selectivity, 22
weasel_selectivity(), *18, 19*
weasel_sensitivity, 23
weasel_subset_to_sentence, 24
weasel_summarize_subset, 25
weasel_summarize_subset(), *16*, *18, 19*, *24, 25*
weasel_summarize_waves, 26
weasel_summarize_waves(), *12, 13*