

Package: masque (via r-universe)

October 19, 2026

Title Structurally Faithful Development Surrogates for Tabular Data

Version 0.15.0

Description Turns a confidential tabular dataset - a single table, a folder of files, or a multi-sheet workbook - into a structurally faithful synthetic clone suitable for pipeline development. Experimental-design allocation and the NA pattern are preserved exactly; categorical design, treatment, and covariate vocabularies can be reversibly aliased; outcome and numeric-covariate values are re-simulated via a Gaussian copula that preserves the global covariance structure. Columns shared across tables are aliased consistently so the synthetic tables still join. A private recipe object round-trips a pipeline written against the synthetic clone onto the original data. Not a differential-privacy or anonymisation tool: outputs are development surrogates, not public-release-safe artefacts.

License MIT + file LICENSE

Encoding UTF-8

Language en-AU

Roxygen list(markdown = TRUE)

Depends R (>= 4.2.0)

Imports cli, data.table, digest, Matrix, S7, stats, utils, withr

Suggests agridat, callr, fst, ggplot2, knitr, maps, readxl, rmarkdown, spelling, testthat (>= 3.0.0), writexl

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://max578.github.io/masque/>, <https://github.com/max578/masque>

BugReports <https://github.com/max578/masque/issues>

Config/roxygen2/version 8.1.0

Repository <https://max578.r-universe.dev>

Date/Publication 2026-10-19 01:49:02 UTC

RemoteUrl <https://github.com/max578/masque>
RemoteRef main
RemoteSha 5156f6189c1116fc6b3b77b1d40c3b043a76a2c3

Contents

apply_recipe	2
audit_mask	4
clean_table	5
conform_table	6
detect_design	8
jitter_coordinates	10
mask	13
mask_set	17
masque	19
plot_design_summary	21
propose_roles	22
read_recipe	25
read_set	26
recipe	27
reveal_maps	28
role_options	28
roles_validate	29
save_recipe	31
set_role	32
synthesise_geospatial	33
synthetic	35
unmask	36
write_set	37
Index	39

apply_recipe	<i>Translate a data frame into the synthetic namespace</i>
--------------	--

Description

Forward translation: takes an original-namespace data frame and returns it renamed and re-labelled to match the synthetic namespace produced by `mask()`. Use this to run a pipeline (trained on the synthetic) against the original data without modifying pipeline code.

Usage

```
apply_recipe(original, rec, check_integrity = TRUE)
```

Arguments

<code>original</code>	A data frame in the original namespace.
<code>rec</code>	A <code>masque_recipe</code> object (e.g. from <code>recipe(m)</code>).
<code>check_integrity</code>	Logical. When TRUE (default), verifies that the NA mask of <code>original</code> matches the recipe's recorded <code>integrity_fp</code> . Mismatches error with guidance. Pass FALSE to bypass when the missingness has legitimately changed since the recipe was built.

Details

Operations applied (in order):

1. **Verify integrity** by comparing the NA mask of `original` to the SHA-256 fingerprint stored on the recipe (controlled by `check_integrity`).
2. **Drop** every column whose resolved action was "drop" on the roles table `mask()` masked with, in either mode.
3. **Subset and reorder** to the columns the recipe knows about.
4. **Re-label factors / characters** for any column with a level map held by the recipe (i.e., treatment and categorical covariates in `collaborate` mode; or treatment in `local` mode with opt-in permutation). Unknown non-NA values fail closed.
5. **Rename columns** per `recipe@column_name_map` when `mask()` was called with `alias_names` (otherwise NULL, and names are unchanged).

Numeric columns are passed through unchanged: the synthetic-namespaces for numeric columns is the same as the original. NA cells in the input remain NA in the output (no synthesis is performed here).

Value

A `data.table` in the synthetic namespace, ready for the pipeline.

See Also

[unmask\(\)](#), [mask\(\)](#).

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
r$role[r$col == "Species"] <- "covariate"
m <- mask(iris, r, mode = "collaborate", seed = 1)
rec <- recipe(m)
iris_in_synth_space <- apply_recipe(iris, rec)
head(iris_in_synth_space)
```

audit_mask

Audit a masque object for leakage and shareability risks

Description

Returns a per-column audit data.table and prints a severity-grouped report. Auto-runs in mode = "collaborate" at `mask()` time and stores the result on the masque object (`m@audit`); for local-mode audits or explicit re-audits, pass the original data frame via `original`.

Usage

```
audit_mask(m, original = NULL, print = TRUE)
```

Arguments

<code>m</code>	A masque object from <code>mask()</code> .
<code>original</code>	Optional. Required when <code>m@audit</code> is NULL (typically in local mode). Used to recompute exact-match-pct etc. on demand.
<code>print</code>	Logical; if TRUE (default), print a styled report.

Details

Each row of the returned data.table holds:

- `col`: column name in the original.
- `role`: assigned role.
- `kind`: storage kind.
- `leakage_class`: low, medium, or high.
- `n_unique_levels`: distinct non-NA values (categorical only).
- `freq_min`: minimum per-level frequency (categorical only).
- `exact_match_pct`: percentage of synthetic cells equal to the original cell (numeric only; cell-by-cell).
- `na_pct`: percentage of NA cells in the original column.
- `na_pattern_uniqueness`: fraction of rows in the original with a globally unique NA pattern (one number per data frame, repeated on every row).
- `alias_status`: aliased, passthrough, or dropped.
- `notes`: short human summary.

Classification heuristics:

- Retained PII-pattern column -> high.
- Treatment unaliased in collaborate -> high.
- Categorical covariate with a frequency-1 level in collaborate -> high.

- Outcome with exact-match-pct > 1\
- Numeric covariate with exact-match-pct > 5\ medium.
- Ignore column retained in local -> low (informational).

As a reference point, a collaborate-mode clone of agridat::besag.met (yield as the outcome, design columns kept) scores low on every column, and none of its yields equals the original value.

Value

The audit data.table, returned invisibly.

See Also

[mask\(\)](#).

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
r$role[r$col == "Species"] <- "covariate"
m <- mask(iris, r, mode = "collaborate", seed = 1)
audit_mask(m)
```

clean_table	<i>Tidy a dirty table's column names and category labels before masking</i>
-------------	---

Description

Real custodian tables arrive with column names that are not valid R names ("Yield (t/ha)", "Site Name"), leading or trailing whitespace in names and factor / character values, and the occasional near-duplicate label ("north" vs "North" vs " north"). clean_table() legalises names and trims whitespace, and reports each fix. Near-duplicate labels are only reported, since merging labels that look alike is a judgement call; conform_table() makes it.

Usage

```
clean_table(df, clean = c("auto", "report", "off"), quiet = FALSE)
```

Arguments

df	A data frame.
clean	One of "auto" (default - legalise names, trim whitespace, report near-duplicates), "report" (legalise names, report what whitespace / near-duplicate changes <i>would</i> be made but apply none), or "off" (legalise names only, skip all other hygiene). Column-name legalisation is applied in every mode – an invalid name silently rewritten downstream corrupts the clone – and is surfaced as a masque_name_repaired warning; only the whitespace and near-duplicate handling is governed by the mode.

`quiet` Logical. When FALSE (default) a cli summary of the fixes and advisories is printed. Set TRUE to suppress it (the report object is returned either way).

Details

The corrections are returned alongside the cleaned data, so `mask()` records them in the recipe and `apply_recipe()` re-applies the same cleaning to the original.

Value

An object of class `masque_cleaning`: a list with

- `data` - the cleaned (or, under `report / off`, unchanged) `data.table`;
- `name_map` - named character `original -> clean` for every column whose name changed (empty if none);
- `level_fixes` - named list, one entry per column whose values were trimmed, each a named character `original -> clean`;
- `near_duplicates` - a `data.table` of report-only label pairs (`col`, `a`, `b`, `kind`) that look like typos but were left untouched;
- `mode` - the clean mode applied.

See Also

`mask()`, `propose_roles()`.

Examples

```
df <- data.frame(
  `Site Name` = c("north ", "north", "South"),
  `Yield (t/ha)` = c(3.1, 2.9, 5.0),
  check.names = FALSE
)
cl <- clean_table(df, quiet = TRUE)
names(cl$data)
cl$near_duplicates
```

<code>conform_table</code>	<i>Bring a cleaned table to an analysable format, reporting every assumption</i>
----------------------------	--

Description

`clean_table()` makes the fixes that are unambiguously safe: it legalises column names and trims whitespace, and it *reports* near-duplicate labels without merging them. `conform_table()` is the next step and makes the two decisions that need judgement: deciding that "NSW" and "Nsw" are one category, and deciding that a character column holding six labels is a factor.

Usage

```
conform_table(
  df,
  merge_labels = c("report", "auto", "off"),
  types = c("report", "auto", "off"),
  max_levels = 20L,
  clean = c("auto", "report", "off"),
  quiet = FALSE
)
```

Arguments

<code>df</code>	A data frame.
<code>merge_labels</code>	One of "report" (default), "auto" or "off". Governs near-duplicate category labels, and treats the two kinds differently. A pair differing only by capitalisation is one category whatever happens, so it is merged onto the more frequent spelling, and on a tie onto whichever appeared first, which is deterministic and recorded. A pair differing by one edit might be two real categories, so it is merged only when one spelling is clearly the more common; on a tie it is refused and reported.
<code>types</code>	One of "report" (default), "auto" or "off". Governs storage format: a low-cardinality character column becomes a factor, a fully numeric character column becomes numeric, and a fully ISO-8601 character column becomes a Date.
<code>max_levels</code>	Cardinality at or below which a character column is treated as categorical. Default 20.
<code>clean</code>	Passed to <code>clean_table()</code> , which always runs first. Default "auto".
<code>quiet</code>	Logical. When FALSE (default) a cli summary is printed.

Details

Both are **off by default**: the default "report" mode lists every change it would make, in plain words, and applies none. Set a mode to "auto" to apply them; the assumption behind each one is recorded in the returned object.

Storage is decided **before** categories, so a column of numbers or dates held as text is typed first and "4.2" is never merged into "4.4". Only columns that remain categorical are considered for merging, and a one-character edit between labels shorter than four characters is ignored ("a" and "b" are one edit apart).

Two labels that differ by an edit are merged only when one spelling is clearly more common; a column is converted only when every value parses; numeric columns keep their storage. Anything not changed for these reasons is reported as a not applied row with the reason.

Value

An object of class `masque_conformance`: a list with

- `data` - the conformed `data.table`;
- `cleaning` - the `masque_cleaning` record from `clean_table()`;

- merges - one row per near-duplicate pair considered (col, from, to, n_from, n_to, applied, reason);
- types - one row per column whose storage was considered (col, from, to, applied, reason);
- assumptions - every decision as a plain sentence (col, assumption, applied), which is the thing to read;
- modes - the modes applied.

See Also

[clean_table\(\)](#) for the safe fixes, [propose_roles\(\)](#) for what a column is, [mask\(\)](#) for the round-trip.

Examples

```
df <- data.frame(
  state = c("NSW", "Nsw", "NSW", "VIC", "VIC"),
  yield = c("3.1", "2.9", "5.0", "4.2", "3.8"),
  sown = c("2024-05-01", "2024-05-03", "2024-05-01", "2024-05-08",
           "2024-05-08"),
  stringsAsFactors = FALSE
)
# report only: nothing is changed
cf <- conform_table(df, quiet = TRUE)
cf$assumptions

# apply the decisions
cf2 <- conform_table(df, merge_labels = "auto", types = "auto", quiet = TRUE)
str(cf2$data)
```

detect_design

Detect environment scope and experimental-design structure

Description

Inspects df and returns an S7 design_summary. Environment scope and experimental-design class are separate conclusions: a table can be a multi-environment trial (MET) even when its within-environment randomisation cannot be recovered from the recorded columns.

Usage

```
detect_design(
  df,
  roles = NULL,
  interactive = FALSE,
  threshold = 0.5,
  tie_delta = 0.02,
  env = NULL
)
```

Arguments

df	A data frame.
roles	Optional roles table (as returned by propose_roles()). When supplied, declared roles constrain candidate generation, and columns roled treatment define the treatment basis.
interactive	If TRUE, when the top-2 rule scores are within tie_delta the user is asked to choose between them via a cli menu. Default FALSE.
threshold	Minimum top-rule score for a class to be reported. Below this, class_label is "none". Default 0.5.
tie_delta	Score difference within which two rules are treated as tied. Default 0.02 – tight enough that 0.05-point score differences (the typical name-bonus / coverage gap) are decisive.
env	Environment specification. NULL performs conservative automatic resolution and leaves ambiguous or weak evidence unresolved. FALSE disables MET handling and runs the legacy whole-table path. A character vector names one or more columns whose interaction defines the environment.

Details

With env = NULL, exact environment names and a bounded set of site-year patterns are assessed conservatively. A site-only candidate auto-resolves only when treatments are replicated across sites. Weak or competing evidence gives an explicit uncertain result, never a guessed single trial. Supply env to define the environment basis, or use env = FALSE to run the pre-0.9 whole-table path exactly.

After the scope step, the pooled legacy detector runs six independent design rules. Each returns a score in [0,1]. The highest-scoring class above threshold is one of "CRD", "RCBD", "IBD/alpha-lattice", "row-column", "split-plot", "factorial", or "none". Ties within tie_delta favour the simpler design. For a MET, per-environment classes, treatment connectivity, and near-disjoint experiment groups are diagnostics only. Dense connectivity calculations that exceed the package safety bound are reported as not_computed. None of these diagnostics proves the original randomisation protocol.

The detector never edits df. Its job is to recommend a role assignment, surface the evidence, and (optionally) draw a sanity check via [plot\(\)](#).

Value

An S7 design_summary object. Legacy design fields include class_label, treatment_col, block_cols, whole_plot_col, sub_plot_col, spatial_cols, scores, evidence, recommended_roles, candidates, and warnings. Scope fields include scope_label, scope_status, scope_confidence, is_met, env_cols, env_method, n_env, group_cols, connectivity, per_env, and within_design_label.

See Also

[propose_roles\(\)](#) for the role table that feeds detection and [plot_design_summary\(\)](#) for the sanity-check visualisation.

Examples

```
# Classic alpha-lattice (24 genotypes, 3 reps, 6 blocks per rep).
if (requireNamespace("agridat", quietly = TRUE)) {
  d <- agridat::john.alpha
  ds <- detect_design(d)
  print(ds)
}

# Observational data frame -> class_label "none".
detect_design(mtcars)

# Explicit two-environment trial.
met <- expand.grid(
  env = factor(c("E1", "E2")),
  rep = factor(seq_len(2L)),
  gen = factor(c("G1", "G2", "G3"))
)
met$yield <- seq_len(nrow(met))
met_design <- detect_design(met, env = "env")
met_design$scope_label
met_design$connectivity$status
```

jitter_coordinates *Coarsen geographic coordinates by an on-land privacy jitter*

Description

Displaces latitude / longitude pairs by a small random distance, in place, so a synthetic table can carry realistic coordinates without revealing the true location of a field, farm or site. Two geographic-masking schemes are offered. The default "donut" displaces every point by a distance drawn uniformly (by area) from an annulus between `min_km` and `max_km`, in a random direction: this guarantees a minimum displacement (a privacy floor – no point is left almost where it started) while bounding the maximum drift. The "gaussian" scheme adds independent normal noise of standard deviation `sd_km` to each axis; it is simpler but has an unbounded tail, so a handful of points can move much further than intended.

Usage

```
jitter_coordinates(
  df,
  lat_col,
  lon_col,
  by = NULL,
  method = c("donut", "gaussian"),
  min_km = 5,
  max_km = 20,
  sd_km = 10,
```

```

    on_land = TRUE,
    max_tries = 100L,
    seed = NULL,
    .group = NULL
  )

```

Arguments

<code>df</code>	A data frame containing the coordinate columns.
<code>lat_col, lon_col</code>	Column names of the latitude and longitude (numeric, in decimal degrees).
<code>by</code>	How rows are grouped into sites. <code>NULL</code> (default) groups rows that share an identical input coordinate pair. A character vector names one or more columns of <code>df</code> whose values identify a site. <code>FALSE</code> displaces every row independently (the pre-0.10.0 behaviour). See the section above.
<code>method</code>	"donut" (default) or "gaussian".
<code>min_km, max_km</code>	Inner and outer radii of the donut, in kilometres (used when <code>method = "donut"</code>). Every site moves at least <code>min_km</code> and at most <code>max_km</code> .
<code>sd_km</code>	Per-axis standard deviation in kilometres (used when <code>method = "gaussian"</code>).
<code>on_land</code>	Controls the land constraint. <code>TRUE</code> (default) rejects any displacement that lands in the sea, using <code>maps::map.where()</code> (requires the <code>maps</code> package). A function <code>function(lon, lat)</code> returning a logical vector supplies your own test (for example an <code>sf</code> -based high-resolution coastline). <code>FALSE</code> applies no constraint.
<code>max_tries</code>	Maximum re-draws per site before giving up. Default 100. A site that cannot be placed on land is set to <code>NA</code> on both axes, with a warning: <code>masque</code> never emits the true coordinate as though it were masked.
<code>seed</code>	Optional integer seed for reproducibility. The caller's RNG state is preserved.
<code>.group</code>	Internal. A pre-computed grouping vector of length <code>nrow(df)</code> , used by <code>mask()</code> to group by the <i>original</i> table's site structure. Takes precedence over <code>by</code> . Not for direct use.

Details

The displacement is computed in kilometres and converted to degrees with a `cos(latitude)` correction on longitude, so the true ground distance matches the requested magnitude at any latitude. Each candidate is checked against a land mask and re-drawn until it falls on land, so a coastal point is never pushed offshore. The original `NA` pattern is preserved cell-by-cell and the two axes stay paired (if either coordinate is missing, both are set to `NA`).

Value

`df`, with `lat_col` and `lon_col` overwritten by the jittered coordinates (rounded to five decimal places, about one metre).

One displacement per site

One displacement is drawn per **site**, not per row, and broadcast to every row of that site. A table that holds many rows per physical place – plots within a trial, samples within a paddock, observations within a farm – therefore comes back with one masked coordinate per place, exactly as the source table has one true coordinate per place. by chooses how a site is identified:

- `by = NULL` (default) treats rows that share an identical input coordinate pair as one site. This needs no user action and cannot alter genuinely point-level data, where every input pair is already distinct.
- `by = c("trial")` (one or more column names) treats rows sharing those values as one site, for the case where a site's recorded coordinates differ slightly between rows. Where such a group holds more than one distinct input coordinate, the group's row-weighted centroid is used and the consolidation is reported.
- `by = FALSE` restores the pre-0.10.0 behaviour of displacing every row independently, and warns if the input carries repeated coordinate pairs. It exists only so a table masked by an earlier version can be reproduced.

Displacing each row of a repeated coordinate separately leaks the site. Donut displacement is isotropic, so the mean of many independent draws around one true site converges on that site: averaging the 360 rows of a single trial recovers the true position to a median 0.58 km, against the 5 km floor the donut was asked for. One draw per site removes the estimator and leaves the full displacement in place.

Repeated rebuilds

One displacement per site holds *within* a masked table. It does not hold across several. Each rebuild with a fresh seed draws independently, so a site masked repeatedly and kept in several versions can be averaged back out, exactly as its rows could be before. Measured at the 5-20 km default:

Rebuilds kept	Median recovery error
1	14.2 km
2	10.1 km
4	6.4 km
8	4.5 km
16	3.1 km

Eight kept rebuilds fall below the 5 km floor the donut was asked for.

The remedy is to **reuse the seed**. The same seed reproduces the same displacement for the same sites, so any number of rebuilds yields one point and nothing to average. Draw a fresh seed only when you intend to supersede every earlier version, and retire the ones you replace.

Choose the unit carefully. It is the finest grouping that denotes **one physical place at one time**. For a multi-year trial series that is location by year, not location alone: trials at one named location in different seasons usually sit in different paddocks, and one displacement for all of them would misplace every season but one.

Choosing the magnitude

The right displacement depends on the density of the entities you are protecting: the masked point should be spatially k-anonymous (roughly, at least k comparable entities lie closer to the masked point than the true one). Individual-level urban health data is typically masked with a standard deviation of about 1 km, because cities are dense. Agricultural fields and farms are orders of magnitude sparser, so a comparable level of protection needs a much larger displacement – a donut of roughly 5 to 20 km (the default) moves a point across several properties while keeping it in the same agroclimatic region. For a formal guarantee, calibrate `min_km` / `max_km` to the local field density to reach a target k-anonymity.

References

- Hampton, K. H., Fitch, M. K., Allshouse, W. B., Doherty, I. A., Gesink, D. C., Leone, P. A., Serre, M. L., & Miller, W. C. (2010). Mapping health data: improved privacy protection with donut method geomasking. *American Journal of Epidemiology*, 172(9), 1062-1069. doi:10.1093/aje/kwq248
- Zandbergen, P. A. (2014). Ensuring confidentiality of geocoded health data: assessing geographic masking strategies for individual-level data. *Advances in Medicine*, 2014, 567049. doi:10.1155/2014/567049

See Also

[synthesise_geospatial\(\)](#) to instead re-anchor coordinates at user-supplied fake centroids.

Examples

```
# Four plots at each of three sites: one masked coordinate per site.
df <- data.frame(
  site = rep(c("A", "B", "C"), each = 4),
  lat = rep(c(-34.9, -35.2, -33.6), each = 4),
  lon = rep(c(138.6, 142.0, 148.2), each = 4)
)
if (requireNamespace("maps", quietly = TRUE)) {
  out <- jitter_coordinates(df, "lat", "lon", min_km = 5, max_km = 20,
    seed = 1)
  nrow(unique(out[c("lat", "lon")])) # 3, one pair per site
}
```

mask

Mask a tabular dataset into a structurally faithful development surrogate

Description

Takes one data frame and a user-edited roles table (from [propose_roles\(\)](#), possibly adjusted with [set_role\(\)](#)) and produces a synthetic clone according to each column's action. Returns a masque S7 object holding the synthetic data and a private masque_recipe.

Usage

```
mask(
  df,
  roles,
  mode = c("local", "collaborate"),
  seed = NULL,
  clean = c("auto", "report", "off"),
  alias_names = FALSE,
  conditional = FALSE,
  ladder = c("hierarchy", "levels"),
  coords = NULL,
  allow_unmasked_coords = FALSE,
  quiet = FALSE,
  .shared_maps = list(),
  ...
)
```

Arguments

<code>df</code>	A data frame, or the <code>masque_conformance</code> object returned by <code>conform_table()</code> , whose data element is then masked.
<code>roles</code>	A roles table from <code>propose_roles()</code> (possibly edited). Tables from <code>masque <= 0.5.0</code> are upgraded with a deprecation warning; see <code>roles_validate()</code> .
<code>mode</code>	Either "local" or "collaborate". When omitted, inherit <code>attr(roles, "mode")</code> , falling back to "local" for a roles table with no mode provenance. An explicit collaborate-to-local downgrade raises a classed <code>masque_mode_downgrade</code> warning.
<code>seed</code>	Optional integer for reproducibility.
<code>clean</code>	Label and column-name hygiene before masking, passed to <code>clean_table()</code> : one of "auto" (default - trim whitespace and report near-duplicate labels), "report" (report only), or "off" (skip). Invalid column names are legalised in every mode – an invalid name silently rewritten during synthesis corrupts the clone – and the repair is raised as a <code>masque_name_repaired</code> warning. The roles table's column references are remapped to the legalised names, the fixes are recorded in the recipe, and <code>apply_recipe()</code> / <code>unmask()</code> reverse them on the round-trip.
<code>alias_names</code>	Hide the column names themselves. FALSE (the default) keeps them. TRUE replaces every retained column name with an opaque alias (<code>col_001</code> , <code>col_002</code> , ... in column order). A character vector names just the columns to alias. The original-to-alias map is stored in the recipe and inverted by <code>apply_recipe()</code> / <code>unmask()</code> , so a pipeline written against the aliased synthetic round-trips. Alias the names when the schema itself is sensitive.
<code>conditional</code>	Logical scalar (default FALSE). The <i>collaborate-grade conditional clone</i> . When FALSE, scrambled numeric columns are re-simulated from one global Gaussian copula - marginals and global covariance survive, but the treatment-to-outcome relationship does not, so a causal model fitted on the clone recovers a null effect. When TRUE, the numeric block is re-simulated <i>within each treatment-by-design</i>

stratum, so a row's synthetic outcome inherits the location of the treatment that row carries. A causal model fitted on the conditional clone recovers the real treatment effect within sampling tolerance. The conditioning columns (treatment plus retained design) are recorded on the recipe.

The stratum is chosen by a **coarsening ladder**. Treatment crossed with every retained design column is the finest rung, but on a replicated factorial that rung holds one row per cell, which is too thin to synthesise. The ladder then drops design columns, in the order set by ladder, until the cells hold at least five rows; treatment columns are never dropped. Whatever remains below that floor is pooled into a global fallback. The rung reached is recorded on the recipe as `conditioning_used`, the columns given up as `conditioning_dropped`, the pooled share as `fallback_frac`, and any coarsening or residual pooling raises a classed `masque_conditional_degraded` warning – including the case where no treatment or design column survives at all, in which case the clone is the pooled copula.

ladder	How the coarsening ladder orders the design columns it may drop, and what it keeps of a dropped one. "hierarchy" (the default) drops plot coordinates (row, col, range, plot) first, then blocking columns, then environment columns (site, year, county, trial, ...), so an environment is never dropped before the replicates inside it; within the last two tiers the column explaining the least variance in the numeric block (adjusted for its number of levels) goes first. The main effect of every dropped column, and of each pair of dropped blocking or environment columns (the replicate inside a county is a different block from the same label in the next county), is estimated by least squares beside the stratum that was kept, removed before the within-stratum copula and added back after it. A clone that conditions on genotype alone therefore still has the original's environment, replicate and block means. Rows pooled into the fallback stratum keep the stratum columns' main effects the same way, which is how a three-replicate variety trial keeps its genotype means. A level with a single row is pooled with the other singletons before the fit, so no row's own outcome is carried. The terms carried are recorded on the recipe as <code>conditioning_shifted</code> . "levels" is the ladder of masque 0.11.1 to 0.12.0: columns are dropped in decreasing order of distinct values and a dropped column leaves no trace in the clone.
coords	Optional geographic-coordinate declaration. Supply one or more latitude/longitude pairs and each is coarsened in place by an on-land jitter (see jitter_coordinates()) instead of being copula-scrambled into implausible locations. A pair is a named vector <code>c(lat = "lat_col", lon = "lon_col")</code> or a named list that also carries jitter parameters (by, method, min_km, max_km, sd_km, on_land); pass several pairs as a list. Defaults to a donut of 5-20 km on land. One displacement is drawn per site and broadcast to that site's rows: by names the columns of <code>df</code> that identify a site, and is omitted when rows sharing an identical input coordinate already identify one. A declared pair always survives masking, coarsened; the recipe records the parameters and the site grouping it was coarsened under, and apply_recipe() retargets to the real coordinates.
allow_unmasked_coords	Single logical. A masked table must not carry a real coordinate, so by default <code>mask()</code> stops when a column whose name says coordinate (gps, lat, lon, easting, ...) would be written through unmasked, and warns when a column

	detected only by the shape of its values would be. State otherwise by declaring the pair to coords (coarsened), giving the column a masking action (drop or scramble), or – having decided the coordinate is not sensitive – setting this to TRUE, which is recorded on the recipe.
quiet	Single logical. TRUE silences the masque_mode_downgrade warning raised when a roles table prepared for "collaborate" is used in "local" mode, for a plan that is deliberately shared across both modes. No other warning is affected; a HIGH-leakage finding is always raised.
.shared_maps	Internal. A named list of pre-computed original -> alias level maps for cross-table linked columns, set by <code>mask_set()</code> . Not for direct use.
...	Must be empty. An unused argument, for example a misspelled name, is an error.

Details

mode = "local" marks the synthetic for owner development only; the reminder is recorded on the recipe and shown when the object prints. mode = "collaborate" additionally jitters re-simulated numerics within their measurement resolution (stochastically rounding integers) and runs `audit_mask()` automatically; a HIGH finding is raised as a classed warning (`masque_high_leakage`) and blocks the package-managed writers (`masque()`'s out, `write_set()`) until it is resolved or explicitly overridden. Which columns are aliased, kept, or dropped is decided by the action column of roles - `propose_roles()` resolves mode-appropriate defaults, so the table you reviewed is the plan that runs.

Collaborate mode changes the transformations and runs the audit. Whether a synthetic table is appropriate for a given collaborator, environment or jurisdiction is a release decision for the data custodian; masque informs that decision, it does not make it.

Value

A masque S7 object. Use `synthetic()` and `recipe()` to extract the components.

Behaviour by action

keep Byte-identical pass-through, both modes.

scramble Numeric outcome / covariate columns are re-simulated jointly via a Gaussian copula on the global Pearson covariance, with empirical-quantile marginals. Categorical, date, and text columns are row-permuted within non-NA positions, class preserved. Treatment columns get a seeded label permutation - the assignment structure never moves.

alias As scramble where applicable, plus opaque label substitution: treatments become `trt_NNN` (`<col>_trt_NNN` when two or more treatment factors are aliased), categorical covariates `<col>_LNNN`, design labels `<col>_DNNN` (in place - structure intact), ids `<col>_INNN` (in place - row linkage intact), text values `<col>_TNNN`.

drop Column excluded from the synthetic, both modes.

The NA mask of every retained column is preserved cell-by-cell. RNG state is preserved across the call.

See Also

[propose_roles\(\)](#), [set_role\(\)](#), [roles_validate\(\)](#), [synthetic\(\)](#), [recipe\(\)](#), [reveal_maps\(\)](#).

Examples

```
r <- propose_roles(iris)
r <- set_role(r, "Sepal.Length", role = "outcome")
m <- mask(iris, r, seed = 1)
head(synthetic(m))
```

mask_set

*Mask a multi-table set with cross-table-consistent aliasing***Description**

The set-level counterpart to [mask\(\)](#). Takes a folder of tabular files, an Excel workbook, or a named list of data frames and produces one synthetic table per input table plus a single private recipe bundle. Columns that are shared across tables - a site code, a genotype name, a plot id appearing in several tables - are aliased *identically everywhere they occur*, so a join written against the synthetic set still resolves on the masked data.

Usage

```
mask_set(
  input,
  roles = NULL,
  links = NULL,
  mode = c("local", "collaborate"),
  seed = NULL,
  clean = c("auto", "report", "off"),
  alias_names = FALSE,
  conditional = FALSE,
  ladder = c("hierarchy", "levels"),
  quiet = FALSE
)
```

Arguments

input	A folder path, an .xlsx / .xls path, or a named list of data frames - anything read_set() accepts.
roles	Optional named list of roles tables, one per input table (names must match). When NULL (default), propose_roles() runs on each table for the chosen mode. Supply edited tables for full control.
links	Optional override for link detection: a character vector of column names to treat as cross-table links, or FALSE to disable linking entirely (each table masked independently). When NULL (default), links are detected automatically.

mode	Either "local" or "collaborate". When omitted and roles are supplied, inherit their common mode. Tables prepared for different modes must be reconciled explicitly. Otherwise default to "local".
seed	Optional integer for reproducibility.
clean	Hygiene mode passed to <code>clean_table()</code> for every table ("auto", "report", or "off").
alias_names	Hide column names. FALSE (default) keeps them; TRUE aliases every non-link column (linked join keys keep their names so the synthetic set stays joinable).
conditional	Logical scalar (default FALSE). Passed through to each per-table <code>mask()</code> call: when TRUE, every table's numeric block is re-simulated within its own treatment-by-design strata so the treatment-to-outcome relationship survives the clone. See <code>mask()</code> for the full account.
ladder	Drop order of the conditioning ladder, "hierarchy" (default) or "levels"; passed through to each <code>mask()</code> call.
quiet	Logical; suppress the link / hygiene report.

Value

A `masque_set` S7 object. Use `synthetic()` for the named list of synthetic tables and `recipe()` for the private recipe bundle.

Links

A *link* is a column name that appears in two or more tables with a compatible kind and overlapping values - the join keys of the set. `mask_set()` proposes links automatically and prints them; pass links to override. Linked columns are aliased in place with a shared map (built over the union of values across all tables), so identical original values map to identical aliases in every table. Set a linked column's action to keep in its roles table to pass it through unmasked instead.

See Also

`mask()`, `read_set()`, `write_set()`.

Examples

```
tables <- list(
  plots = data.frame(
    site = c("A", "A", "B", "B"),
    gen = c("x", "y", "x", "y"),
    yield = c(3.1, 2.9, 4.0, 3.7)
  ),
  sites = data.frame(
    site = c("A", "B"),
    rainfall = c(420, 560)
  )
)
m <- mask_set(tables, mode = "collaborate", seed = 1, quiet = TRUE)
synthetic(m)$plots$site
synthetic(m)$sites$site # same aliases -> the join still works
```

masque

*Mask a dataset end to end with one guided call***Description**

The front door. `masque()` walks the whole procedure - read the data, propose column roles, (in an interactive session) pause for you to review the plan, mask, audit, and optionally write the result - from a single call. It dispatches on the input: a single file or data frame goes through `mask()`; a folder, an Excel workbook, or a named list of tables goes through `mask_set()`.

Usage

```
masque(
  input,
  roles = NULL,
  out = NULL,
  mode = c("local", "collaborate"),
  seed = NULL,
  clean = c("auto", "report", "off"),
  alias_names = FALSE,
  conditional = FALSE,
  ladder = c("hierarchy", "levels"),
  ask = interactive(),
  overwrite = FALSE,
  quiet = FALSE,
  allow_high = FALSE
)
```

Arguments

<code>input</code>	A data frame, a single tabular file (<code>.csv</code> / <code>.tsv</code> / <code>.fst</code>), a folder of such files, an Excel workbook, or a named list of data frames.
<code>roles</code>	Optional. A roles table (single-table input) or named list of roles tables (set input). When supplied, the interactive review is skipped.
<code>out</code>	Optional output path. For a single table, a <code>.csv</code> file (or <code>.xlsx</code> with <code>writexl</code>). For a set, a folder (one CSV per table) or an <code>.xlsx</code> workbook. When <code>NULL</code> (default) nothing is written.
<code>mode</code>	Either <code>"local"</code> (default) or <code>"collaborate"</code> .
<code>seed</code>	Optional integer for reproducibility.
<code>clean</code>	Hygiene mode passed to <code>clean_table()</code> (<code>"auto"</code> , <code>"report"</code> , <code>"off"</code>).
<code>alias_names</code>	Hide column names; see <code>mask()</code> / <code>mask_set()</code> .
<code>conditional</code>	Logical scalar (default <code>FALSE</code>). The conditional clone mode passed through to <code>mask()</code> / <code>mask_set()</code> : when <code>TRUE</code> , numeric columns are re-simulated within each treatment-by-design stratum so the treatment-to-outcome relationship survives the clone. See <code>mask()</code> for the full account.

ladder	Drop order of the conditioning ladder, "hierarchy" (default) or "levels"; passed through to <code>mask()</code> / <code>mask_set()</code> .
ask	Whether to pause for interactive review when roles is not supplied. Defaults to <code>interactive()</code> . Set FALSE to proceed with the proposed plan without prompting.
overwrite	Passed to the writer when out is set.
quiet	Suppress progress messages. Warnings - including the HIGH-leakage finding - are never suppressed.
allow_high	Logical (default FALSE). When out is set and the collaborate-mode audit flagged HIGH leakage, the write is refused. Pass TRUE to write anyway after your own review; the override raises a <code>masque_high_override</code> warning and is recorded in the recipe's warnings, so the exception stays auditable.

Details

It is also fully scriptable. Pass an edited roles table (or named list of them) to skip the interactive review, and an out path to write the masked result in one go. The returned object is the same `masque` / `masque_set` you would get from the lower-level verbs, so anything you can do with those you can do with the result here.

Value

A `masque` object (single-table input) or a `masque_set` object (set input), invisibly. Use `synthetic()` and `recipe()`.

The guided flow

1. **Read** the input into one or more clean rectangular tables.
2. **Propose roles** for every column (skipped if you pass roles).
3. **Review** - in an interactive session with no roles supplied, the proposed plan is printed and you are asked to proceed, edit, or stop. Editing opens the roles table in the spreadsheet editor (`utils::edit()`) where the platform provides one; when it cannot start (for example macOS without XQuartz, or a headless session), a console editor takes over - pick a column, then a role and an action from numbered menus, with every change applied through `set_role()` so default actions re-resolve exactly as on the scriptable path. With `ask = FALSE` (the default in non-interactive use) the proposed plan is used as-is, with a note.
4. **Mask** the data in the chosen mode.
5. **Audit** - in collaborate mode the leakage audit runs and its headline is printed. A HIGH finding surfaces as a classed warning (`masque_high_leakage`) - it is never silenced by the guided flow.
6. **Write** - if out is set, the masked data is written there (mirroring the input format), *unless* the audit left unresolved HIGH findings: then the write is refused (nothing is written) and the flagged columns are listed. Resolve them and mask again, or pass `allow_high = TRUE` after your own review - the override is warned (`masque_high_override`) and recorded on the recipe. The private recipe is never written automatically; persist it yourself with `save_recipe()`.

See Also

[mask\(\)](#), [mask_set\(\)](#), [propose_roles\(\)](#), [set_role\(\)](#), [write_set\(\)](#), [save_recipe\(\)](#).

Examples

```
# Scripted single-table use (no prompt because roles are supplied):
r <- propose_roles(iris)
r <- set_role(r, "Sepal.Length", role = "outcome")
m <- masque(iris, roles = r, seed = 1, ask = FALSE, quiet = TRUE)
head(synthetic(m))
```

plot_design_summary *Sanity-check visualisation for detected scope and design*

Description

Plots the structure that drove the [detect_design\(\)](#) verdict. A detected MET uses a compact environment-coverage overview by default. Set environment to draw one selected environment. A single-trial panel depends on the detected design class:

Usage

```
plot_design_summary(
  x,
  df,
  engine = c("base", "ggplot2"),
  environment = NULL,
  ...
)
```

Arguments

x	A design_summary object from detect_design() .
df	The data frame that was passed to detect_design() . It is used to draw replication tiles, spatial layouts, and the NA-pattern. The data are deliberately not stored in the summary.
engine	"base" (default) or "ggplot2". The latter requires ggplot2 and falls back to "base" with a warning if unavailable.
environment	Optional single environment label. For a detected MET, the default is a compact all-environment overview. Supplying a label draws the selected environment's field layout or legacy design diagnostic.
...	Ignored.

Details

- CRD, factorial, none -> frequency-of-treatment + NA-pattern.
- RCBD, IBD/alpha-lattice -> treatment x block replication tile.
- row-column -> spatial layout tile (row x col, fill = treatment).
- split-plot -> factor-nesting tree + within-block replication.

Output is purely diagnostic. Do not use it as a publication figure (use `desplot::desplot()` or `ggplot2`-based packages for that).

Value

With `engine = "ggplot2"`, the `ggplot` object, so it can be modified or saved with `ggplot2::ggsave()`; nothing is drawn until it is printed. With `engine = "base"`, the input `x` invisibly, after the plot has been drawn.

Examples

```
if (requireNamespace("agridat", quietly = TRUE)) {
  d <- agridat::john.alpha
  ds <- detect_design(d)
  plot(ds, df = d)
}

met <- expand.grid(
  env = factor(c("E1", "E2")),
  rep = factor(seq_len(2L)),
  gen = factor(c("G1", "G2", "G3"))
)
met$yield <- seq_len(nrow(met))
met_design <- detect_design(met, env = "env")
plot(met_design, df = met)
```

propose_roles

Propose role and action classifications for the columns of a data frame

Description

Generates a heuristic two-axis classification for every column of `df`: a role (what the column *is*) and an action (what `mask()` will *do* to it). The user is expected to inspect this table and edit it - directly or via `set_role()` - before passing it to `mask()`.

Usage

```
propose_roles(df, mode = c("local", "collaborate"), detect = TRUE)
```

Arguments

df	A data frame. Must have at least one column.
mode	The masking mode the table is being prepared for: "local" (default) or "collaborate". Stored as attr(roles, "mode") and used to resolve default actions.
detect	Logical scalar (default TRUE). When TRUE, run <code>detect_design()</code> and overlay its recommended role hints.

Value

A data.table with one row per column: col, role, action, kind (storage kind: numeric, integer, factor, character, logical, date, datetime, other), freq_or_range, pii_suspected, and notes. The target mode is stored as attr(roles, "mode").

The two axes

role describes the column and determines the *mechanics* of any synthesis:

design Experimental / structural columns (site, block, rep, plot, year). Mechanics of alias: labels are substituted in place, structure intact.

treatment Assignment columns (variety, genotype, dose). Labels are remapped in place - the assignment structure never moves. scramble = seeded label permutation; alias = opaque labels (trt_001).

outcome Numeric response columns. scramble re-simulates via the Gaussian copula, jointly with scrambled numeric covariates. Multiple outcomes are supported.

covariate Everything measured alongside. Numeric: copula re-simulation. Categorical: row permutation, plus opaque label aliasing under alias.

date Date / POSIX / difftime columns. scramble row- permutes within the observed values; class and NA pattern are preserved.

id Identifier columns. Never scrambled (that would break row linkage); alias substitutes opaque per-value labels in place, preserving linkage.

text Free-text columns. scramble row-permutes; alias tokenises each distinct string.

other Classes masque cannot synthesise (list columns, exotic S4, ...). Keep or drop only.

action sets the masking depth per column:

keep Byte-identical pass-through, both modes.

scramble Re-simulate (numeric) or row-permute (categorical / date / text); original label vocabulary remains visible.

alias Scramble where applicable, plus opaque label substitution - the vocabulary itself is hidden.

drop Column excluded from the synthetic, both modes.

The proposed action column is resolved for mode, so the table you edit shows the actual masking plan. Re-assigning a column's role with `set_role()` re-resolves its default action; a direct `roles$role[...] <- ...` edit leaves action untouched (set it to NA to have `mask()` re-resolve the default).

Default classification rules, applied in order

1. PII-pattern column names (contact, email, phone, gps, latitude / longitude, postcode, ssn, password, owner, farmer, operator, etc., case-insensitive substring) -> pii_suspected = TRUE and action drop in **both** modes. Re-role deliberately if the column must survive.
2. Date / POSIXct / POSIXlt / difftime columns -> role date, action scramble (row permutation).
3. ID-pattern names (\\bid\\b, _id\$, ^id_) -> role id; kept in local mode, dropped in collaborate mode.
4. Design-pattern names (rep, block, row, col(umn)?, range, plot(no)?, site, env(ironment)?, trial, year, season, colrep, tos) -> role design, action keep.
5. Treatment-pattern names (treatment, variety, cultivar, genotype, ^trt, ^dose) -> role treatment; kept in local mode, aliased in collaborate mode.
6. Character columns with > 50% unique values on non-NA -> role text; kept in local mode, dropped in collaborate mode.
7. Unsupported classes -> role other, action keep, with a note.
8. Everything else -> role covariate, action scramble. Re-role response variables as outcome.

No outcome is required: with no column roled outcome, the copula simply re-simulates all scrambled numeric columns jointly.

Since masque 0.3.0, `propose_roles()` also calls `detect_design()` by default (detect = TRUE) and applies the detected design's recommended_roles on top of the name-based heuristic, re-resolving the default action for any promoted column. The design summary is stashed as `attr(roles, "design")`. Pass detect = FALSE for the name-only heuristic.

Multi-environment trials

Since masque 0.9.1, high-confidence environment columns detected by `detect_design()` are promoted to role design. In local mode their values remain byte-identical. In collaborate mode categorical environment labels default to `alias`, which preserves row assignment and factor codes while hiding the vocabulary. Numeric environment columns default to `keep` and raise a `masque_environment_disclosure` warning for explicit review.

Weak or competing environment candidates never change roles automatically. Inspect `attr(roles, "design")` and use `set_role()` when domain knowledge should override an uncertain result. Preserving environment allocation does not imply that synthesised outcomes preserve treatment-by-environment effects. An explicitly chosen action is pinned and is not overwritten by a later design-role promotion.

See Also

`set_role()` to edit; `roles_validate()` for the fail-closed validation applied at `mask()` time.

Examples

```
propose_roles(iris)
propose_roles(iris, mode = "collaborate")

met <- expand.grid(
```

```

  env = factor(c("E1", "E2")),
  rep = factor(seq_len(2L)),
  gen = factor(c("G1", "G2", "G3"))
)
propose_roles(met, mode = "collaborate")[, c("col", "role", "action")]

```

read_recipe

Read a masque recipe from disk

Description

Loads a recipe written by [save_recipe\(\)](#). Validates that the file contains a `masque_recipe` and informs (does not error) if the recipe was written by a different package version than the one currently installed.

Usage

```
read_recipe(path)
```

Arguments

`path` File path.

Value

A `masque_recipe` object.

See Also

[save_recipe\(\)](#), [recipe\(\)](#).

Examples

```

r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
m <- mask(iris, r, mode = "collaborate", seed = 1)
tmp <- tempfile(fileext = ".rds")
save_recipe(recipe(m), tmp)
rec2 <- read_recipe(tmp)
rec2

```

read_set

*Read a set of tables from a folder, an Excel workbook, or a list***Description**

Ingests a multi-table dataset into a named list of data frames, ready for `mask_set()`. Three input shapes are accepted:

Usage

```
read_set(input, sheets = NULL, pattern = NULL)
```

Arguments

input	A folder path, an .xlsx / .xls path, or a named list of data frames.
sheets	For an Excel workbook, an optional character vector of sheet names to read (default: all sheets).
pattern	For a folder, an optional regular expression to select files (default: all .csv / .tsv / .fst).

Details

a folder path Every .csv, .tsv, and .fst file in the folder becomes one table, named by its file name (without extension). CSV / TSV are read with `data.table::fread()`; .fst needs the Suggested `fst` package.

an Excel workbook (.xlsx / .xls) Every sheet becomes one table, named by its sheet name. Needs the Suggested `readxl` package.

a named list of data frames Returned as-is after validation.

Only clean rectangular tables are supported. A sheet or file that does not read as a rectangle - missing header row (auto-named . . . 1 columns), zero rows, or zero columns - fails with an explanatory error naming the offending table. `masque` does not attempt to recover merged cells, multi-row headers, or junk rows; tidy the source first.

Value

A named list of `data.tables`.

See Also

`mask_set()`, `write_set()`.

Examples

```
tables <- list(
  plots = data.frame(site = c("A", "B"), yield = c(3.1, 4.2)),
  sites = data.frame(site = c("A", "B"), rainfall = c(400, 550))
)
read_set(tables)
```

recipe

Extract the recipe from a masque object

Description

The recipe is **private**: at least as sensitive as the original data. Never share alongside the synthetic. By default `print(recipe(m))` redacts all level maps; use [reveal_maps\(\)](#) for an explicit reveal.

Usage

```
recipe(m)
```

Arguments

`m` A masque object from [mask\(\)](#), or a masque_set from [mask_set\(\)](#).

Value

A `masque_recipe`, or for a set a `masque_recipe_set` bundle.

See Also

[synthetic\(\)](#), [reveal_maps\(\)](#), [mask\(\)](#), [mask_set\(\)](#).

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
m <- mask(iris, r, seed = 1)
recipe(m)
```

 reveal_maps

Reveal the level maps held inside a recipe

Description

Explicit, audited reveal: prints each original -> synthetic level map held by the recipe. **Use sparingly.** Recipe maps are at least as sensitive as the original data; printing them defeats the redaction built into `print()` and `summary()`.

Usage

```
reveal_maps(rec)
```

Arguments

`rec` A masque_recipe object (e.g., from `recipe(m)`).

Value

`rec`, invisibly.

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
r$role[r$col == "Species"] <- "covariate"
m <- mask(iris, r, mode = "collaborate", seed = 1)
rec <- recipe(m)
reveal_maps(rec)
```

 role_options

List every role and action combination masque accepts

Description

Renders the two-axis vocabulary as a table: every role paired with every action, the storage kinds the pair works for, and the reason the pair is constrained when it is. The table is generated from the same compatibility rules `roles_validate()` applies at `mask()` time, so what it shows is exactly what a roles table is allowed to say.

Usage

```
role_options(kind = NULL)
```

Arguments

kind Optional storage kind to filter for: one of "numeric", "integer", "factor", "character", "logical", "date", "datetime", or "other". When supplied, only the combinations workable for a column of that kind are returned. The default NULL returns the full grid.

Details

A column's kind is never chosen: `propose_roles()` derives it from the column's class. To move a column to a different kind, convert the column in the data and re-propose.

Value

A data.table with one row per role-action pair and four columns: role, action, kinds (the storage kinds the pair is valid for: "all", "none", "all except other", or a comma-separated list), and notes (why the pair is constrained, empty when it is not).

See Also

`propose_roles()`, `set_role()`, `roles_validate()`.

Examples

```
role_options()
role_options(kind = "factor")
subset(role_options(), role == "design")
```

roles_validate	<i>Validate a roles table</i>
----------------	-------------------------------

Description

Fail-closed validation of a two-axis roles table before `mask()` consumes it. Returns the validated table - with any NA actions resolved to their (role, kind, mode) defaults - so callers can use the return value directly.

Usage

```
roles_validate(roles, df = NULL, mode = NULL)
```

Arguments

roles A roles table from `propose_roles()` (possibly edited), or a v1 roles table (deprecated, upgraded with a warning).

df Optional data frame. If supplied, roles is checked for one-to-one column-name correspondence with df.

mode Optional mode ("local" or "collaborate") used to resolve NA actions and to upgrade v1 tables. Defaults to `attr(roles, "mode")`, falling back to "local".

Details

Tables produced by masque 0.5.0 and earlier (no action column; v1 vocabulary with keep / ignore roles and the optional mask_levels column) are upgraded in place with a deprecation warning. The upgrade preserves the v1 semantics exactly: v1 keep becomes action keep; v1 ignore becomes role id / text / other with action keep in local mode and drop in collaborate mode; v1 treatment mask_levels = "permute" becomes action scramble.

Hard errors:

- missing required columns (col, role, action, kind);
- unknown role (not in design, treatment, outcome, covariate, date, id, text, other) or unknown action (not in keep, scramble, alias, drop);
- any NA role (an NA *action* is allowed - it resolves to the default for the row's role and kind);
- an incompatible (role, action, kind) combination, e.g. design + scramble, numeric + alias, id + scramble, other + anything but keep / drop;
- duplicate col entries;
- if df supplied: any df column missing from roles, or any roles column missing from df.

Loud advisories (warnings, not errors):

- every action is keep - the "synthetic" would equal the original byte-for-byte;
- the table was proposed for one mode but is being validated for another (actions are taken as-is; defaults are not re-resolved).

Value

The validated (and possibly upgraded / resolved) roles table, invisibly.

See Also

[propose_roles\(\)](#), [set_role\(\)](#).

Examples

```
r <- propose_roles(iris)
r <- set_role(r, "Sepal.Length", role = "outcome")
roles_validate(r, iris)
```

save_recipe	<i>Save a masque recipe to disk</i>
-------------	-------------------------------------

Description

Writes the recipe to a single .rds file. The default is **runtime-minimal**: no simulator state (copula covariance, raw margins) is written, only the translation maps, factor metadata, storage classes, integrity fingerprint, and warnings. This keeps the saved artefact small and reduces the information that would leak if the recipe file alone were shared.

Usage

```
save_recipe(rec, path, include_simulator = FALSE)
```

Arguments

rec	A masque_recipe object, e.g. from recipe(m).
path	File path. By convention, .rds extension.
include_simulator	Logical. Has no effect; the recipe is always written without simulator state.

Details

include_simulator = TRUE is accepted and has no effect: the recipe does not carry simulator state.

Recipes are at least as sensitive as the original data. Protect the saved file at the same security class as the original. Note that save_recipe() writes plain R serialisation - it does not encrypt. The saved recipe is a re-identification key: store it under your organisation's access controls and key-management practice, not alongside the synthetic output.

Value

path, invisibly.

See Also

[read_recipe\(\)](#), [recipe\(\)](#).

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
m <- mask(iris, r, mode = "collaborate", seed = 1)
tmp <- tempfile(fileext = ".rds")
save_recipe(recipe(m), tmp)
rec2 <- read_recipe(tmp)
```

set_role

Set the role and action of one or more columns in a roles table

Description

Ergonomic editor for the two-axis roles table returned by `propose_roles()`. Setting a role without an explicit action re-resolves the action to the default for the new role, the column's kind, and the mode the table was proposed for – so a re-roled column never silently carries a stale action from its previous role.

Usage

```
set_role(roles, cols, role = NULL, action = NULL)
```

Arguments

roles	A roles table from <code>propose_roles()</code> (possibly edited).
cols	Character vector of column names to edit. Every entry must be present in <code>roles\$col</code> .
role	Optional single role to assign to all of cols. One of "design", "treatment", "outcome", "covariate", "date", "id", "text", "other".
action	Optional single action to assign to all of cols. One of "keep", "scramble", "alias", "drop". When NULL (the default) and role was supplied, the action is re-resolved to the default for the new role.

Details

Direct edits (`roles$role[roles$col == "x"] <- "outcome"`) also work, but leave `roles$action` unchanged.

Value

The edited roles table.

See Also

`propose_roles()`, `roles_validate()`.

Examples

```
r <- propose_roles(iris)
r <- set_role(r, "Sepal.Length", role = "outcome")
r <- set_role(r, "Species", action = "keep")
r[, c("col", "role", "action")]
```

`synthesise_geospatial` *Re-anchor synthetic geospatial coordinates at plausible-but-fake locations*

Description

Replaces the latitude / longitude values in a masked data frame with coordinates anchored at user-supplied centroids and clustered to preserve the original's site-count-per-anchor structure. The function never reads the real coordinates beyond counting how many distinct sites the original holds per anchor level – so it leaks the replication-per-site distribution and the count of distinct sites, nothing more.

Usage

```
synthesise_geospatial(
  synth,
  original,
  anchor_col,
  lat_col,
  lon_col,
  anchor_centroids,
  site_spread_deg = 0.6,
  jitter_deg = 0.05,
  seed = NULL
)
```

Arguments

<code>synth</code>	A synthetic data frame (typically <code>synthetic(mask(...))</code>).
<code>original</code>	The original data frame from which <code>synth</code> was derived (needed only to count distinct sites per anchor).
<code>anchor_col</code>	Name of the column whose levels anchor each cluster (e.g., "M_STATE"). Must exist in both <code>synth</code> and <code>original</code> .
<code>lat_col, lon_col</code>	Column names of the latitude and longitude fields to overwrite in <code>synth</code> .
<code>anchor_centroids</code>	Named list keyed by anchor levels; each element is a length-2 numeric named <code>c(lat, lon)</code> . The user supplies plausible centroids (e.g., state centroids); the function never infers them from the original to avoid leaking position information. Any other shape (a data frame, an unnamed list, an element that is not a <code>c(lat, lon)</code> pair, or a list naming none of the anchor levels) is refused with a <code>masque_bad_anchor_centroids_refusal</code> error. A level absent from an otherwise well-formed list is warned about and its rows get NA.
<code>site_spread_deg</code>	Half-width of the box (in decimal degrees) around each anchor centroid within which fake site centroids are uniformly placed. Default 0.6.

jitter_deg	Within-site uniform jitter (in decimal degrees) added to each row's assigned site centroid. Default 0.05.
seed	Optional integer seed for reproducibility. The function uses <code>withr::local_preserve_seed()</code> so the caller's RNG state is left untouched.

Details

Typical use: after `mask()` produces a synthetic with copula-drawn or missing coordinates, call `synthesise_geospatial()` to substitute plausible points. The synthetic ends up with:

the same number of distinct sites per anchor level (e.g., if the original has five distinct trial sites in NSW, the synthetic will have five fake sites in NSW);

the original's per-site replication distribution (each fake site receives a share of the synthetic rows proportional to its real counterpart's count);

within-site tight clustering and between-site spread (small jitter within site; larger spread between sites within each anchor centroid's neighbourhood).

What the function does **not** preserve:

- the real positions of the sites (they are random within a user-defined neighbourhood of each anchor centroid);
- the relative spacing or bearings between real sites;
- any spatial autocorrelation in the outcome.

Coordinates that are NA in the original remain NA in the synthetic – the NA pattern is preserved cell-by-cell.

Value

`synth`, with `lat_col` and `lon_col` overwritten by the re-anchored coordinates.

Examples

```
# Toy example: 50 rows split across two states.
set.seed(1)
n <- 50
df <- data.frame(
  state = sample(c("NSW", "VIC"), n, replace = TRUE),
  lat   = stats::rnorm(n, -33, 0.3),
  lon   = stats::rnorm(n, 145, 0.3),
  y     = stats::rnorm(n)
)
roles <- propose_roles(df, detect = FALSE)
roles$role[roles$col == "y"] <- "outcome"
roles$role[roles$col %in% c("lat", "lon")] <- "covariate"
roles$role[roles$col == "state"] <- "design"
m <- mask(df, roles, mode = "collaborate", seed = 1L)
centroids <- list(
  NSW = c(lat = -32.5, lon = 147),
  VIC = c(lat = -36.5, lon = 144)
```

```

)
synth_geo <- synthesise_geospatial(
  synthetic(m), df,
  anchor_col = "state", lat_col = "lat", lon_col = "lon",
  anchor_centroids = centroids, seed = 2L
)
head(synth_geo[, c("state", "lat", "lon")])

```

synthetic

Extract the synthetic data from a masque object

Description

Extract the synthetic data from a masque object

Usage

```
synthetic(m)
```

Arguments

m A masque object from `mask()`, or a masque_set from `mask_set()`.

Value

For a masque, a data.table (the synthetic table); for a masque_set, a named list of synthetic tables.

See Also

`recipe()`, `mask()`, `mask_set()`.

Examples

```

r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
m <- mask(iris, r, seed = 1)
head(synthetic(m))

```

unmask

*Translate data from the synthetic namespace back to the original***Description**

Inverse of `apply_recipe()`. Accepts a data frame or an atomic vector. For an atomic factor / character vector with a recipe that holds multiple level maps, column must name which map to invert. Atomic numeric, integer, logical, and Date / POSIXct vectors are returned unchanged (no inverse map applies – these are pass-through under `apply_recipe()` too).

Usage

```
unmask(x, rec, column = NULL)
```

Arguments

x	A data frame or an atomic vector to translate from synthetic-namespace to original-namespace.
rec	A <code>masque_recipe</code> object.
column	Optional column name. Only consulted for atomic factor / character x when the recipe holds more than one level map. Ignored for atomic numeric / logical / Date-like input (pass-through), but if supplied is validated against the recipe's known columns.

Details

The most common pattern is round-tripping pipeline predictions:

```
fit          <- my_model(synthetic(m))          # train on synthetic
orig_in_synth_space <- apply_recipe(original, recipe(m)) # forward
preds_synth  <- predict(fit, orig_in_synth_space)
preds_orig   <- unmask(preds_synth, recipe(m))    # inverse
```

An unknown level (a synthetic alias not in the recipe's map) is an error; it is never coerced to NA.

Value

An object of the same type as x, in the original namespace.

See Also

`apply_recipe()`, `mask()`.

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Species"] <- "treatment"
m <- mask(iris, r, mode = "collaborate", seed = 1)
synth <- synthetic(m)
rec <- recipe(m)
# round-trip a synthetic-namespace value back to its original label
head(unmask(synth$Species, rec, column = "Species"))
```

write_set

Write a masked set to disk, mirroring the input format

Description

Writes the synthetic tables of a `mask_set()` result to disk. The output format mirrors the request: a `.xlsx` path produces one workbook with one sheet per table; a folder path produces one `.csv` per table. The private recipe bundle is **never** written by this function - persist it separately with `save_recipe()` and protect it at the same security class as the original data.

Usage

```
write_set(m, path, overwrite = FALSE, allow_high = FALSE)
```

Arguments

<code>m</code>	A <code>masque_set</code> object from <code>mask_set()</code> .
<code>path</code>	Output location. A path ending in <code>.xlsx</code> writes a workbook (needs the <code>Suggested writexl</code> package); any other path is treated as a folder, created if necessary, and receives one <code><table>.csv</code> per table.
<code>overwrite</code>	Logical. When <code>FALSE</code> (default), writing over an existing file or a non-empty folder errors.
<code>allow_high</code>	Logical (default <code>FALSE</code>). Override the <code>HIGH</code> -leakage write refusal after your own review; the override is raised as a <code>masque_high_override</code> warning.

Details

In collaborate mode the mask-time audit gates the write: unresolved `HIGH` leakage findings refuse the write (nothing is written) until the flagged columns are re-roled, aliased, or dropped - or the refusal is explicitly overridden with `allow_high = TRUE`, which raises a `masque_high_override` warning so the exception stays visible.

Value

`path`, invisibly.

See Also

[mask_set\(\)](#), [read_set\(\)](#), [save_recipe\(\)](#).

Examples

```
tables <- list(
  plots = data.frame(site = c("A", "B"), yield = c(3.1, 4.2)),
  sites = data.frame(site = c("A", "B"), rain = c(420, 560))
)
m <- mask_set(tables, seed = 1, quiet = TRUE)
dir <- file.path(tempdir(), "masked_set")
write_set(m, dir)
list.files(dir)
```

Index

`apply_recipe`, 2
`apply_recipe()`, 6, 14, 15, 36
`audit_mask`, 4
`audit_mask()`, 16

`clean_table`, 5
`clean_table()`, 6–8, 14, 18, 19
`conform_table`, 6
`conform_table()`, 5, 14

`detect_design`, 8
`detect_design()`, 21, 23, 24

`interactive()`, 20

`jitter_coordinates`, 10
`jitter_coordinates()`, 15

`maps::map.where()`, 11
`mask`, 13
`mask()`, 2–6, 8, 11, 17–23, 27–29, 35, 36
`mask_set`, 17
`mask_set()`, 16, 19–21, 26, 27, 35, 37, 38
`masque`, 19
`masque()`, 16

`plot()`, 9
`plot_design_summary`, 21
`plot_design_summary()`, 9
`print()`, 28
`propose_roles`, 22
`propose_roles()`, 6, 8, 9, 13, 14, 16, 17, 21, 24, 29, 30, 32

`read_recipe`, 25
`read_recipe()`, 31
`read_set`, 26
`read_set()`, 17, 18, 38
`recipe`, 27
`recipe()`, 16–18, 20, 25, 31, 35
`reveal_maps`, 28

`reveal_maps()`, 17, 27
`role_options`, 28
`roles_validate`, 29
`roles_validate()`, 14, 17, 24, 28, 29, 32

`save_recipe`, 31
`save_recipe()`, 20, 21, 25, 37, 38
`set_role`, 32
`set_role()`, 13, 17, 20–24, 29, 30
`summary()`, 28
`synthesise_geospatial`, 33
`synthesise_geospatial()`, 13
`synthetic`, 35
`synthetic()`, 16–18, 20, 27

`unmask`, 36
`unmask()`, 3, 14
`utils::edit()`, 20

`write_set`, 37
`write_set()`, 16, 18, 21, 26