Package: masque (via r-universe)

May 31, 2026

Title Structurally Faithful Development Surrogates for Tabular Data

Version 0.4.1

Description Turns a single tabular dataset into a structurally faithful synthetic clone suitable for pipeline development. Experimental-design columns and the NA pattern are preserved exactly; treatment and categorical-covariate level vocabularies are optionally aliased; outcome and numeric-covariate values are re-simulated via a Gaussian copula that preserves the global covariance structure. 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)

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports cli, digest, MASS, Matrix, S7, stats, tibble, utils, withr

Suggests agridat, fst, ggplot2, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/max578/masque

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

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

Date/Publication 2026-05-31 11:03:39 UTC

RemoteUrl https://github.com/max578/masque

RemoteRef main

RemoteSha 386576ffd0bbb8bb3debb6f20342b523f7f7e374

Contents

`apply_recipe` *Translate a data frame into the synthetic namespace*

Description

Forward translation: takes an `original`-namespace data frame and returns it renamed and relabelled 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

`original`	A data frame in the original namespace.
`rec`	A `masque_recipe` object (e.g. from `recipe(m)`).
`check_integrity`	Logical. When `TRUE` (default), verifies that the NA mask of `original` matches the recipe's recorded `integrity_fp`. 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** columns that `mask()` dropped (in `collaborate` mode this is every `ignore` column; in `local` mode no columns are dropped).
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` (currently `NULL`; reserved for a future opt-in column-aliasing flag — see `vignette("roadmap")`).

Numeric columns are passed through unchanged: the synthetic-namespace 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 tibble 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 tibble 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

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

Details

Each row of the returned tibble 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 (CODEX-aligned):

- 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).

Step 7 will lower numeric exact-match-pct under collaborate by adding within-resolution jitter; until then, expect medium leakage on collaborate-mode numerics.

Value

The audit tibble, 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)
```

---

detect_design *Detect the experimental-design structure of a data frame*

Description

Inspects df and returns an S7 design_summary describing the most likely experimental design — one of "CRD", "RCBD", "IBD/alpha-lattice", "row-column", "split-plot", "factorial", or "none" (observational / no detectable design).

Usage

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

Arguments

df	A data frame.
roles	Optional roles tibble (as returned by propose_roles()). When supplied, columns roled outcome / ignore are excluded from factor candidates, and any column roled treatment is forced as the working treatment.
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.

Details

Detection runs six independent rules; each returns a score in $[0, 1]$. The orchestrator picks the highest-scoring class above `threshold`. Ties within `tie_delta` are broken in favour of the simpler design (CRD < RCBD < factorial < IBD < row-column < split-plot).

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 with slots `class_label`, `treatment_col`, `block_cols`, `whole_plot_col`, `sub_plot_col`, `spatial_cols`, `scores`, `evidence`, `recommended_roles`, `candidates`, `warnings`.

See Also

`propose_roles()` for the role tibble that feeds detection; `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)
```

---

mask *Mask a tabular dataset into a structurally faithful development surrogate*

Description

Takes one data frame and a user-edited `roles` tibble (from `propose_roles()`) and produces a synthetic clone whose experimental design and NA pattern are preserved, while outcome and numeric-covariate values are re-simulated via a Gaussian copula and categorical-covariate values are row-permuted. Returns a `masque` S7 object holding the synthetic data and a private `masque_recipe`.

Usage

```
mask(df, roles, mode = c("local", "collaborate"), seed = NULL, ...)
```

Arguments

df	A data frame.
roles	A tibble produced by propose_roles() (possibly edited). May optionally include a mask_levels column ("permute" enables local-mode seeded permutation on the treatment column).
mode	Either "local" (default) or "collaborate".
seed	Optional integer for reproducibility.
...	Currently ignored.

Details

mode = "local" keeps original column / level vocabularies and warns that the synthetic is for owner development only. mode = "collaborate" opaque-aliases treatment and categorical-covariate level vocabularies (trt_001, <col>_L01) and drops ignore columns; the resulting synthetic can be passed to a collaborator while the recipe stays private. In collaborate mode, numeric draws are jittered within their measurement resolution, integer columns are stochastically rounded, and audit_mask() runs automatically.

Value

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

Behaviour by role

design Byte-identical pass-through.

treatment Local: pass-through (optional opt-in seeded permutation via roles$mask_levels = "permute"). Collaborate: opaque alias trt_NNN.

outcome + **numeric** covariate Re-simulated jointly via a Gaussian copula on global Pearson covariance. Empirical-quantile marginals (type 1: returns observed values).

categorical covariate Row-permuted within non-NA positions. Local: vocabulary preserved. Collaborate: opaque alias <col>_LNN.

ignore Local: passes through. Collaborate: dropped.

RNG state is preserved across the call.

See Also

propose_roles(), roles_validate(), synthetic(), recipe(), reveal_maps().

Examples

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

`plot_design_summary` *Sanity-check visualisation for a detected design*

Description

Plots the structure that drove the `detect_design()` verdict. The panel layout depends on the detected class:

Usage

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

Arguments

`x`	A `design_summary` object from `detect_design()`.
`df`	The data frame that was passed to `detect_design()` — used to draw replication tiles, spatial layouts, and the NA-pattern. Required for every class except `"none"` with no factors.
`engine`	`"base"` (default) or `"ggplot2"`. The latter requires ggplot2; falls back to `"base"` with a warning if unavailable.
`...`	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

The input `x`, invisibly. Called for the plot side-effect.

Examples

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

propose_roles *Propose role classifications for the columns of a data frame*

Description

Generates a heuristic role tibble for every column of `df`. The user is expected to inspect this tibble and edit it before passing it to `mask()`. Heuristics are seeds, not law.

Usage

```
propose_roles(df, detect = TRUE)
```

Arguments

`df`	A data frame. Must have at least one column.
`detect`	Logical scalar (default `TRUE`). When `TRUE`, run `detect_design()` and overlay its recommended role hints on the name-based heuristic. Stash the `design_summary` as `attr(roles, "design")`. When `FALSE`, only the v0.2.x name-based heuristic runs.

Details

Roles are exactly one of:

`design` Byte-identical pass-through. Trial / site / replicate / block / plot / row / column / year etc.

`treatment` Same factor cardinality and per-level frequency; optional label aliasing or seeded permutation.

`outcome` Re-simulated via Gaussian copula. Multiple allowed.

`covariate` Numeric: Gaussian copula (joint with outcomes). Categorical: row-permuted, levels preserved (local) or aliased (collaborate).

`ignore` Dropped or passed through depending on `mask()` options; auto-assigned for date/time, free text, and PII-pattern names.

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) -> `ignore` with `pii_suspected = TRUE`.
2. Date / POSIXct / POSIXlt / difftime columns -> `ignore`.
3. ID-pattern names (`\\bid\\b`, `_id$`, `^id_`) -> `ignore`.
4. Design-pattern names (`rep`, `block`, `row`, `col(umn)?`, `range`, `plot(no)?`, `site`, `env(ironment)?`, `trial`, `year`, `season`, `colrep`, `tos`) -> `design`.
5. Treatment-pattern names (`treatment`, `variety`, `cultivar`, `genotype`, `^trt`, `^dose`) -> `treatment`.
6. Character columns with > 50% unique values on non-NA -> `ignore` (likely free text).

7. Everything else -> `covariate`. The user re-classifies one or more columns as `outcome`.

Failing to designate at least one `outcome` is a hard error at `mask()` time (via `roles_validate()`).

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. This promotes structurally-identified block / treatment columns even when the column names do not match the design / treatment regexes. The resulting design summary is stashed as `attr(roles, "design")` so the user can `plot()` it or inspect alternates. Pass `detect = FALSE` to recover the v0.2.x name-only behaviour byte-for-byte.

Value

A tibble with one row per column, containing:

- `col`: column name.
- `role`: one of `design`, `treatment`, `outcome`, `covariate`, `ignore`.
- `kind`: storage kind (`numeric`, `integer`, `factor`, `character`, `logical`, `date`, `datetime`, `other`).
- `freq_or_range`: brief summary string (range for numeric, level count for factor, etc.).
- `pii_suspected`: `TRUE` if column name matches a PII pattern.
- `notes`: short explanation of the auto-classification.

See Also

`roles_validate()` for the fail-closed validation applied at `mask()` time.

Examples

```
propose_roles(iris)
```

---

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()`.

`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 returned by `mask()`.

Value

A `masque_recipe` S7 object.

See Also

`synthetic()`, `reveal_maps()`, `mask()`.

Examples

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

roles_validate *Validate a roles tibble*

Description

Fail-closed validation of a `roles` tibble before `mask()` consumes it. Errors are raised for every misuse the v0.2 spec calls out.

Usage

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

Arguments

`roles`	A tibble produced by `propose_roles()` (possibly edited).
`df`	Optional data frame. If supplied, `roles` is checked for one-to-one column-name correspondence with `df`.

Details

Hard errors:

- missing required columns (`col`, `role`, `kind`);
- unknown role string (not in `c("design","treatment","outcome","covariate","ignore")`);
- any `NA` role;
- zero columns flagged `outcome`;
- more than one column flagged `treatment` (joint-treatment masking is not yet supported by `mask()`);
- duplicate `col` entries;
- if `df` supplied: any `df` column missing from `roles`, or any `roles` column missing from `df`.

Returns the validated `roles` invisibly (mirrors `stopifnot`-style use).

Value

`roles`, invisibly.

See Also

`propose_roles()`.

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
roles_validate(r, iris)
```

---

save_recipe *Save a masque recipe to disk*

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. Reserved for a future release. Currently a no-op (recipe is always written runtime-minimal).

Details

include_simulator = TRUE is accepted but is currently a no-op: the recipe does not carry simulator state. The flag is reserved for a future release that will let read_recipe() regenerate fresh synthetic samples without access to the original data (see vignette("roadmap")).

Recipes are at least as sensitive as the original data. Protect the saved file at the same security class as the original.

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)
```

---

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

Description

Replaces the latitude / longitude values in a masqued 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

`synth`	A synthetic data frame (typically `synthetic(mask(...))`).
`original`	The original data frame from which `synth` was derived (needed only to count distinct sites per anchor).
`anchor_col`	Name of the column whose levels anchor each cluster (e.g., `"M_STATE"`). Must exist in both `synth` and `original`.
`lat_col, lon_col`	Column names of the latitude and longitude fields to overwrite in `synth`.
`anchor_centroids`	Named list keyed by anchor levels; each element is a length-2 numeric named `c(lat, lon)`. The user supplies plausible centroids (e.g., state centroids); the function never infers them from the original to avoid leaking position information.
`site_spread_deg`	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 `withr::local_preserve_seed()` 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 returned by mask().

Value

A tibble: the synthetic data frame.

See Also

recipe(), mask().

Examples

```
r <- propose_roles(iris)
r$role[r$col == "Sepal.Length"] <- "outcome"
m <- suppressWarnings(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 masque_recipe 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
```

Unknown levels (synthetic aliases not in the recipe's map) fail closed with an informative error rather than silently coercing to NA.

Value

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

See Also

apply_recipe(), mask().

Index