Package: dataganger (via r-universe)

July 22, 2026

Title Synthetic Data Doubles for Safer Prototyping

Version 0.7.0

Author Lennon Li [aut, cre]

Maintainer Lennon Li <yeli@biostats.ai>

Description Creates synthetic data doubles from real datasets for prototyping, teaching, 'shiny' development, and AI-assisted programming. Provides data profiling, role detection, configurable synthesis, utility comparison, and disclosure-risk warnings. Synthetic outputs are intended to reduce direct disclosure risk, not to guarantee privacy.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports cli, digest, dplyr, haven, jsonlite, readr, readxl, rlang, stats, tibble, utils, withr, yaml, zip

Suggests bslib, callr, DT, ggplot2, knitr, nnet, pkgdown, plotly, rmarkdown, shiny, shinytest2, spelling, synthpop, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://dataganger.biostats.ai/, https://github.com/lennon-li/dataganger

BugReports https://github.com/lennon-li/dataganger/issues

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

LazyData true

VignetteBuilder knitr

Config/pak/sysreqs make libx11-dev zlib1g-dev

Repository https://lennon-li.r-universe.dev

Date/Publication 2026-07-22 23:26:04 UTC

RemoteUrl https://github.com/lennon-li/dataganger

RemoteRef HEAD

RemoteSha e6bd4126386bae2125382b23e53eff6223ff97a7

Contents

assess_kanonymity . . . 2
check_code_readiness . . . 3
compare_synthetic . . . 4
dataganger_cli . . . 5
detect_roles . . . 5
enforce_kanon . . . 6
example_admin_claims . . . 7
example_health_survey . . . 8
example_registry . . . 9
export_diagnostic_package . . . 9
export_synthetic . . . 10
geographic_sample . . . 12
individual_sample . . . 13
looks_aggregated . . . 13
make_agent_bundle . . . 14
plot_comparison . . . 15
privacy_check . . . 15
profile_data . . . 16
read_input . . . 17
report_issue . . . 18
run_app . . . 19
suggest_min_rows . . . 19
synth_spec . . . 21
synthesize_data . . . 23
temporal_sample . . . 24

Index **25**

`assess_kanonymity` *Assess k-anonymity over a set of quasi-identifier columns*

Description

Cross-tabulates the quasi-identifier columns and reports how many records fall in combinations (equivalence classes) smaller than `k`. `NA` is treated as a distinct level so that missing values cannot mask a small cell.

Usage

```
assess_kanonymity(data, qi_cols, k = 5)
```

Arguments

data	A data frame.
qi_cols	Character vector of quasi-identifier column names.
k	Minimum acceptable cell size (default 5).

Value

A list with no_qi (logical), smallest_cell (integer), n_below, pct_below, and worst_cells (a tibble of the smallest combinations and their counts).

---

check_code_readiness *Check whether synthetic data is code-compatible with the original*

Description

Evaluates whether code written against the synthetic development twin will run against the original data without errors. Checks column presence, R class compatibility, factor level coverage, all-NA columns, zero-variance columns, missingness spikes, and ID uniqueness. haven_labelled columns currently round-trip as character in synthetic data, so that class change is expected for now.

Usage

```
check_code_readiness(original, synthetic, roles = NULL)
```

Arguments

original	The original data frame.
synthetic	The synthetic data frame (from synthesize_data()).
roles	Optional; a dataganger_roles object from detect_roles(). Used for ID-uniqueness checks. Computed internally if NULL.

Value

An S3 object of class dataganger_code_readiness with components:

checks A tibble with one row per check: check, scope, column, status ("pass"/"warn"/"fail"), message.

summary List with n_pass, n_warn, n_fail, ready (TRUE when n_fail == 0).

meta List with dimensions and generated_at.

Examples

```
orig <- data.frame(x = 1:10, y = factor(letters[1:10]))
spec <- synth_spec(purpose = "demo")
syn  <- synthesize_data(orig, spec)
check_code_readiness(orig, syn)
```

`compare_synthetic` *Compare original and synthetic datasets*

Description

Compares an original dataset with its synthetic double across dataset-level dimensions, numeric distributions, categorical distributions, and numeric correlations. Returns a structured `dataganger_comparison` object.

Usage

```
compare_synthetic(original, synthetic, roles = NULL)
```

Arguments

`original`	The original data frame.
`synthetic`	The synthetic data frame (from `synthesize_data()`).
`roles`	Optional; a `dataganger_roles` object from `detect_roles()`.

Value

An S3 object of class `dataganger_comparison`, a list with components `dataset`, `numeric`, `categorical`, `relationship`, `interaction`, `privacy_flags`, and `meta`.

Examples

```
dat <- data.frame(x = 1:10, y = letters[1:10])
spec <- synth_spec(purpose = "demo")
syn <- synthesize_data(dat, spec)
compare_synthetic(dat, syn)
```

dataganger_cli *DataGangeR command-line interface*

Description

Testable command-line entrypoint used by the installed `exec/dataganger` shim.

Usage

```
dataganger_cli(args = commandArgs(trailingOnly = TRUE), quit = FALSE)
```

Arguments

`args`	Character vector of trailing command-line arguments.
`quit`	Logical. When `TRUE`, terminate the R process using the returned status code. Tests pass `FALSE` and assert the integer code.

Value

Integer status code: `0` success, `1` processing error, `2` syntax error.

detect_roles *Detect data roles for each column*

Description

Applies heuristic-based role detection to every column in a data frame. Roles include a recommended synthesis role plus the two primary disclosure axes used by the Configure step: whether a column points to a person (`identifies`) and whether it is sensitive. The legacy single `disclosure_role` value is retained as derived compatibility metadata for existing synthesis/export/CLI paths.

Usage

```
detect_roles(data, profile = NULL)
```

Arguments

`data`	A data frame.
`profile`	Optional; a `dataganger_profile` object from `profile_data()`. If `NULL` (the default), profiling is performed internally.

Value

An S3 object of class `dataganger_roles`, a tibble with columns:

variable Column name.

class R class of the column.

recommended_role Role detected by heuristic.

user_role User-supplied override (initially `NA`).

simulation How the column is treated during synthesis.

reason Justification for the recommended role.

identifies Whether the column points to a person: `"none"`, `"combination"`, or `"direct"`.

sensitive Logical flag for whether the column is sensitive if revealed.

user_identifies User-supplied override for `identifies` (initially `NA`).

user_sensitive User-supplied override for `sensitive` (initially `NA`).

disclosure_role Disclosure role. `NA` (unselected) is the conservative default whenever detection is not confident; the user must choose a role before generating. `"direct"` and `"sensitive"` are the only values auto-assigned (confident identifier / known-sensitive name). `"quasi"` and `"none"` are user-assigned choices only.

disclosure_reason Justification for the auto-assigned disclosure role.

Examples

```
df <- data.frame(
  id   = 1:50,
  date = as.Date("2020-01-01") + 0:49,
  city = rep(c("Toronto", "Vancouver", "Montreal"), length.out = 50),
  cat  = factor(rep(letters[1:3], length.out = 50))
)
detect_roles(df)
```

`enforce_kanon` *Enforce k-anonymity on a synthetic dataset (output guarantee)*

Description

Shapes the synthetic output so that no quasi-identifier combination appears in fewer than `k` records. Direct identifiers are removed. Quasi-identifiers are coarsened step-by-step and any residual cell still below `k` has its QI values blanked (`NA`). Operates on the output only.

Usage

```
enforce_kanon(synthetic, roles, k = 5, max_steps = 6L, max_suppress_frac = 0.2)
```

Arguments

synthetic	A synthetic data frame.
roles	A roles object/data frame with variable + disclosure_role.
k	Minimum cell size (default 5).
max_steps	Maximum coarsening iterations (default 6).
max_suppress_frac	Feasibility backstop. If satisfying k over the quasi-identifier set would require blanking more than this fraction of rows, k-anonymity is treated as infeasible for the chosen quasi-identifier (QI) set: the coarsening and suppression steps are *not* applied, the synthetic output is returned populated, and a warning explains that no k-anonymity protection was applied to that output. Default 0.2.

Value

The shaped synthetic data frame, with an attribute kanon recording the achieved state (smallest_cell, suppressed_cells, suppressed_rows, suppressed_row_frac, qi_cols, k, infeasible). suppressed_rows/suppressed
count actual blanked rows across the QI columns – distinct from suppressed_cells, which counts the number of distinct QI combinations folded into suppression. The two can differ a lot: reaching k can require absorbing a few whole neighbouring cells (suppression works at cell granularity, not row granularity), and a handful of small cells sitting next to one dominant cell can end up suppressing most or all of a QI column even though only a few original cells were actually below k.

---

example_admin_claims *Example administrative claims dataset*

Description

A realistic-but-fictional administrative claims dataset with 300 synthetic records. Contains claim identifiers, procedure codes (haven-labelled), costs, and provider locations. No real patient data.

Usage

```
example_admin_claims
```

Format

A tibble with 300 rows and 9 columns:

claim_id Integer. Claim identifier.

patient_id Character. Patient identifier.

service_date Date. Date of service.

dx_code Factor. Diagnosis code.

proc_code haven_labelled. Procedure type (Consult / Surgery / Lab / Imaging).

cost Numeric. Claim cost in dollars, some missing.

approved Logical. Whether the claim was approved, some missing.

provider_city Character. City of the service provider.

postal_code Character. Forward sortation area (FSA).

`example_health_survey` *Example health survey dataset*

Description

A realistic-but-fictional health survey dataset with 200 synthetic records. Contains demographics, clinical measures, and haven-labelled smoking status. No real patient data.

Usage

```
example_health_survey
```

Format

A tibble with 200 rows and 10 columns:

record_id Character. Record identifier.

age Numeric. Age in years.

sex Factor. Biological sex (Male / Female).

bmi Numeric. Body mass index, with some missing values.

smoking_status haven_labelled. Smoking status (Current / Former / Never).

systolic_bp Numeric. Systolic blood pressure, some missing.

diastolic_bp Numeric. Diastolic blood pressure.

survey_date Date. Date of survey response.

province Factor. Canadian province abbreviation.

comments Character. Free-text comments, some missing.

example_registry *Example disease registry dataset*

Description

A realistic-but-fictional disease registry dataset with 150 synthetic records. Contains enrollment data, disease staging (haven-labelled), biomarker values, and patient status. No real patient data.

Usage

```
example_registry
```

Format

A tibble with 150 rows and 10 columns:

subject_id Character. Subject identifier.

enroll_date Date. Date of enrollment.

age_at_enroll Numeric. Age at enrollment in years.

disease_stage haven_labelled. Disease stage (Stage I-IV).

biomarker_a Numeric. Biomarker A level, some missing.

biomarker_b Numeric. Biomarker B level, some missing.

status Factor. Current patient status.

last_visit Date. Date of last follow-up visit, some missing.

region Factor. Geographic region.

notes Character. Clinical notes, some missing.

export_diagnostic_package

Export a Lens-compatible diagnostic schema for a dataset

Description

Profiles a data frame and writes a `diagnostic_view.json` describing column roles, sensitivity, and exposure levels. Does not synthesise data. Intended for agent pre-inspection and Lens ingestion.

Usage

```
export_diagnostic_package(
  data,
  path,
  roles = NULL,
  profile = NULL,
  overwrite = FALSE
)
```

Arguments

data	A data frame to describe.
path	Output path for the JSON file.
roles	Optional; a dataganger_roles object from detect_roles(). Computed internally if NULL.
profile	Optional; a dataganger_profile object from profile_data(). Computed internally if NULL.
overwrite	Logical. When FALSE (the default), aborts if path already exists.

Value

Invisibly, the written JSON path.

Examples

```
dat <- data.frame(age = c(34, 29, 41), grp = c("a", "b", "c"))
export_diagnostic_package(dat, path = tempfile(fileext = ".json"))
```

---

export_synthetic *Export a synthetic data bundle*

Description

Writes a reviewable export bundle containing the synthetic data, a human guide, an optional comparison report, a combined agent recipe, the packaged agent instructions, and a manifest. By default the bundle is written as a zip archive.

Usage

```
export_synthetic(
  synthetic,
  original = NULL,
  comparison = NULL,
  privacy = NULL,
  path,
  format = c("zip", "dir"),
  sanitize_for_spreadsheets = TRUE,
  purpose = NULL,
  roles = NULL,
  include_original_names = NULL,
  fail_on_exact_match = FALSE,
  include_report = TRUE,
  kanon_acknowledged = FALSE,
  include_dictionary = TRUE,
  code_readiness = NULL,
  compact = FALSE,
  overwrite = FALSE
)
```

Arguments

synthetic	A synthetic data frame, typically from synthesize_data().
original	Optional original data frame. When provided, used for the data dictionary, comparison fallback, privacy fallback, and exact-row guard.
comparison	Optional dataganger_comparison object. If NULL and original is supplied, compare_synthetic() is run automatically.
privacy	Optional dataganger_privacy_check object. If NULL and original is supplied, privacy_check() is run automatically at the post stage.
path	Output path. Required. For format = "zip", this is the archive path. For format = "dir", this is the output directory.
format	Character. One of "zip" or "dir".
sanitize_for_spreadsheets	Logical. When TRUE (the default), character-like cells beginning with =, +, -, or @ after leading whitespace are prefixed with a single quote before CSV export.
purpose	Optional purpose label for README text. Defaults to the purpose recorded in attr(synthetic, "spec") when available.
roles	Optional recorded role decisions as a dataganger_roles data frame. When supplied, the export bundle includes the exact column decisions needed to reproduce the same synthetic output.
include_original_names	Logical or NULL. Controls whether the human guide and manifest recipe preserve original variable names. When NULL, this defaults to TRUE unless name_strategy = "dictionary_only", in which case it defaults to FALSE.
fail_on_exact_match	Logical. When TRUE, abort export if exact-row matches are detected for nrow(original) >= 20. When FALSE (the default), exact-row matches are recorded in the privacy report and manifest, and a warning is emitted instead.
include_report	Logical. When TRUE (the default), write human/comparison_report.html. If rmarkdown/knitr are unavailable, the report is skipped with a message instead of an error.
kanon_acknowledged	Logical. Records whether a human explicitly acknowledged exporting a bundle whose k-anonymity backstop was infeasible. Interactive UI export uses this to clear the bundle blocker; non-interactive exports leave it FALSE.
include_dictionary	Deprecated, ignored.
code_readiness	Optional dataganger_code_readiness object from check_code_readiness(). When supplied, writes agent/code_readiness_report.json into the bundle.
compact	Deprecated, ignored.
overwrite	Logical. When FALSE (the default), existing output paths are refused.

Value

Invisibly, the written bundle path.

Examples

```
dat <- data.frame(
  age = 21:70,
  score = seq(10, 59),
  grp = rep(LETTERS[1:5], each = 10)
)
spec <- synth_spec(purpose = "demo", seed = 1, engine = "internal")
syn <- synthesize_data(dat, spec)
export_synthetic(syn, path = tempfile(fileext = ".zip"), include_report = FALSE)
```

---

geographic_sample *Geographic synthetic sample data*

Description

A synthetically generated dataset of 50 regional summary records for use as sample input in the DataGangeR Shiny app. Simulates public-health surveillance data aggregated by region. Generated with `set.seed(42)`.

Usage

```
geographic_sample
```

Format

A data frame with 50 rows and 5 columns:

region Region identifier (Region_01 through Region_50)

population Regional population count

rate_per_100k Event rate per 100,000 population

category Area classification (Urban / Suburban / Rural)

risk_level Assigned risk level (Low / Medium / High)

Source

Synthetically generated via `data-raw/Geographic_sample.R`

individual_sample *Individual-level synthetic sample data*

Description

A synthetically generated dataset of 200 individual records for use as sample input in the DataGangeR Shiny app. Contains demographic and health variables with realistic distributions. Generated with `set.seed(42)`.

Usage

```
individual_sample
```

Format

A data frame with 200 rows and 7 columns:

id Integer record identifier

age Age in years (18–85)

sex Sex (Male / Female / Other)

income Annual income in dollars (log-normal, some NAs)

education Highest education level

smoker Logical smoking status

bmi Body mass index

Source

Synthetically generated via `data-raw/individual_sample.R`

looks_aggregated *Heuristic: does this data frame look pre-aggregated (a table of counts)?*

Description

Disclosure control assumes individual-level microdata. A positive result should drive a non-blocking warning, not a separate policy.

Usage

```
looks_aggregated(data)
```

Arguments

`data`	A data frame.

Value

A list with `aggregated` (logical) and `reason` (character).

`make_agent_bundle` *Create a one-command agent-ready bundle from a raw data file*

Description

Reads a data file, profiles it, detects column roles, synthesizes data, and exports a zip bundle suitable for passing to an AI agent.

Usage

```
make_agent_bundle(
  file,
  out,
  purpose = "development",
  seed = NULL,
  overwrite = FALSE,
  ...
)
```

Arguments

`file`	Path to the input data file. Passed to `read_input()`.
`out`	Path for the output zip file.
`purpose`	Synthesis purpose preset. Defaults to `"development"`. See `synth_spec()` for valid values.
`seed`	Optional integer random seed for reproducible synthesis.
`overwrite`	Logical. When `FALSE` (the default), aborts if `out` already exists.
`...`	Additional arguments passed to `read_input()` only (e.g. `encoding`, `sheet`).

Value

Invisibly, the written bundle path.

Examples

```
dat <- data.frame(
  age = c(24, 28, 35, 41),
  score = c(88, 91, 84, 95),
  grp = c("A", "A", "B", "B")
)
path <- tempfile(fileext = ".csv")
readr::write_csv(dat, path)
make_agent_bundle(
```

```
  file = path,
  out = tempfile(fileext = ".zip"),
  purpose = "demo",
  seed = 1
)
```

`plot_comparison` *Plot comparison summaries*

Description

Produces two bar charts comparing original and synthetic data: standardized differences for numeric columns and total variation distance for categorical columns. Requires `ggplot2` (Suggests).

Usage

```
plot_comparison(comparison)
```

Arguments

`comparison`	A `dataganger_comparison` object from `compare_synthetic()`.

Value

Invisibly, a list with two `ggplot` objects: `numeric` and `categorical`. Each is `NULL` if no columns of that type exist.

Examples

```
dat <- data.frame(x = 1:10, y = letters[1:10])
spec <- synth_spec(purpose = "demo")
syn <- synthesize_data(dat, spec)
cmp <- compare_synthetic(dat, syn)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_comparison(cmp)
}
```

`privacy_check` *Run disclosure-risk privacy checks*

Description

Scans original and (optionally) synthetic data for disclosure-risk flags. Supports two stages: `"pre"` (before synthesis, requires only the original dataset and roles) and `"post"` (after synthesis, requires both original and synthetic).

Usage

```
privacy_check(
  original,
  synthetic = NULL,
  roles = NULL,
  stage = c("pre", "post"),
  spec = NULL
)
```

Arguments

original	The original data frame.
synthetic	Optional; the synthetic data frame (required for stage = "post").
roles	Optional; a dataganger_roles object from detect_roles(). Recommended for pre-stage flag detection. When omitted, fallback name/type heuristics are used.
stage	Character. "pre" or "post".
spec	Optional; a dataganger_spec object. When provided at stage = "post", cross-checks that synthesis parameters were applied (e.g. date coarsening, ID removal).

Value

An S3 object of class dataganger_privacy_check, a tibble with columns variable, flag, severity, stage, and recommendation.

Examples

```
df <- data.frame(id = 1:50, x = rnorm(50), city = rep("Toronto", 50))
roles <- detect_roles(df)
privacy_check(df, roles = roles, stage = "pre")
```

---

profile_data *Profile a dataset column-by-column*

Description

Profiles each column in a data frame, detecting type, computing summary statistics, missingness, cardinality, and flags for free-text, dates, and haven-labelled vectors.

Usage

```
profile_data(data)
```

Arguments

data	A data frame or tibble.

Value

An S3 object of class `dataganger_profile`, which is a list containing:

- `profile`: a tibble with one row per column.
- `n_rows`: total number of rows.
- `n_cols`: total number of columns.
- `generated_at`: POSIXct timestamp of when profiling ran.

Examples

```
df <- data.frame(
  id = 1:5,
  name = letters[1:5],
  score = c(10.1, 15.2, NA, 13.8, 11.0)
)
profile_data(df)
```

---

read_input *Read a data file into a tibble*

Description

Reads CSV, Excel (`.xlsx`, `.xls`), SAS (`.sas7bdat`), and XPT (`.xpt`) files into a tibble. Dispatches on file extension.

Usage

```
read_input(file, sheet = NULL, encoding = NULL, col_select = NULL, ...)
```

Arguments

`file`	Path to the data file.
`sheet`	For Excel files, the sheet name or index to read. Passed to `readxl::read_excel()`. Ignored for non-Excel formats.
`encoding`	Character encoding for CSV files (e.g. `"UTF-8"`, `"latin1"`). Passed as `readr::locale(encoding = encoding)`. Ignored when reading non-CSV formats or when the caller already supplies a `locale` argument in `...`.
`col_select`	Optional character vector of column names to keep. For CSV files the selection is passed directly to `readr::read_csv()` so excluded columns are never parsed. For other formats columns are subset after reading.
`...`	Additional arguments passed to the underlying reader (`readr::read_csv()`, `readxl::read_excel()`, `haven::read_sas()`, or `haven::read_xpt()`).

Value

A `tibble::tibble()`. SAS/XPT imports preserve `haven_labelled` vectors as-is.

Examples

```
path <- tempfile(fileext = ".csv")
readr::write_csv(data.frame(id = 1:3, grp = c("a", "b", "c")), path)
read_input(path)
```

`report_issue` *Report a problem or share feedback*

Description

Prints a pre-filled GitHub issue you can copy into your browser for the `lennon-li/dataganger` repository, with package and R environment details already populated. Use this to report a bug, suggest a feature, or send general feedback without copying session details by hand.

Usage

```
report_issue(
  message = NULL,
  context = NULL,
  type = c("feedback", "bug", "feature")
)
```

Arguments

`message`	Character. A short description of the problem or suggestion. If `NULL`, a placeholder prompt is used.
`context`	Character. Optional context about where the issue happened, such as `"Shiny app"` or `"export_synthetic()"`.
`type`	Character. One of `"feedback"`, `"bug"`, or `"feature"`.

Value

Invisibly, the GitHub issue URL that was printed.

Examples

```
if (interactive()) {
  report_issue(
    message = "The export step was unclear when I skipped the dictionary.",
    context = "Shiny app",
    type = "feedback"
  )
}
```

run_app *Launch the DataGangeR Shiny Application*

Description

Opens the DataGangeR interactive workflow in a local Shiny app. Requires the `shiny`, `bslib`, `DT`, and `plotly` packages.

Usage

```
run_app(max_upload_mb = 50, launch = interactive(), port = NULL, ...)
```

Arguments

`max_upload_mb`	Maximum file upload size in megabytes. Default 50.
`launch`	Whether to launch the app. Default `interactive()`. Set to `FALSE` to configure options without blocking (useful for testing).
`port`	Port to pass to `shiny::runApp()`. Default `NULL`.
`...`	Additional arguments passed to `shiny::runApp()`.

Value

Invisibly `NULL`.

Examples

```
if (interactive()) {
  run_app()
}
```

suggest_min_rows *Suggest a sufficient synthetic row count*

Description

Given a `profile_data()` profile (which carries cross-column coverage information), suggests how many rows to synthesize so that the synthetic data can still represent every category combination and every category level observed in the original data, without blindly matching a large original row count.

Usage

```
suggest_min_rows(
  profile,
  roles = NULL,
  data = NULL,
  k = 5L,
  threshold = 1000L,
  cap = 5000L
)
```

Arguments

profile	A dataganger_profile from profile_data().
roles	Optional; a dataganger_roles object. When provided together with data, the coverage computation is filtered to only the columns whose effective role is synthesizable (excludes alphanumeric IDs, free text, and user-excluded columns).
data	Optional; the original data frame. When provided alongside roles, coverage is recomputed on the filtered column subset so that the suggestion reacts to role changes on the Configure page.
k	Reserved for a future k-anonymity-style cell-size floor; unused by the current coverage rule.
threshold	Row count at or above which a reduction is suggested.
cap	Maximum suggested row count from combination coverage.

Details

The rule (coverage-based) is:

- For small inputs (fewer than threshold rows, default 1000) the original row count is kept — there is nothing to gain from reducing.
- Otherwise the suggestion is the number of observed cross-column category combinations, capped at cap (default 5000) to avoid suggesting millions of rows on wide data, and floored at the largest per-column distinct count so every level remains representable. The suggestion never exceeds the original row count.

Continuous columns are covered by preserving their min/max (already handled by the synthesis engine); they do not raise the suggested count.

Value

A list with:

n Suggested integer row count.

rationale Human-readable explanation.

original_n Original row count.

combination_count Observed category-combination count (or NA).

floor Per-column distinct floor used (or NA).

capped TRUE if the cap bound the suggestion.

reduced TRUE if the suggestion is below the original count.

Examples

```
p <- profile_data(datasets::iris)
suggest_min_rows(p)
```

---

synth_spec *Create a synthesis specification*

Description

Builds a synthesis specification from a purpose preset with optional user overrides. The specification records the synthesis parameters but does not check engine availability - that is done by synthesize_data().

Usage

```
synth_spec(
  purpose,
  level = NULL,
  n = NULL,
  roles = NULL,
  privacy = NULL,
  name_strategy = NULL,
  seed = NULL,
  engine = NULL,
  acknowledge_risk = FALSE,
  ...
)
```

Arguments

purpose	Character. A single non-missing string: "demo", "development", or "analytics".
level	Character or NULL. Synthesis level: "schema" or "marginal". If NULL, derived from the preset.
n	Integer or NULL. Number of rows to synthesize. If NULL, defaults to nrow(original) at synthesis time.
roles	A dataganger_roles object or NULL. Column role assignments.
privacy	A dataganger_privacy_check object or NULL. When stage == "pre", flags harden defaults (e.g. IDs dropped, free text removed).
name_strategy	Character or NULL. How synthetic column names are handled: "preserve" keeps your original column names, "generic" replaces them with neutral names (col_1, col_2, ...), and "dictionary_only" anonymizes the names but records the mapping in the exported data dictionary. If NULL, derived from the preset.

seed — Integer or NULL. Reproducibility seed. Fixes the random draw so the same spec and data reproduce the exact same synthetic output.

engine — Character or NULL. Optional explicit synthesis engine: "auto" clears any explicit engine choice so synthesize_data() derives the engine from the objective, "internal"/"marginal" synthesizes each column from its own distribution (fast, dependency-free, ignores cross-column relationships), and "synthpop" models columns conditionally so correlations and joint structure are preserved (higher fidelity, needs the synthpop package). If NULL, synthesize_data() derives the engine from the objective.

acknowledge_risk — Logical. Required to be TRUE when purpose = "analytics".

... — Additional decision parameters passed to the spec list. These are the same settings exposed under *Synthesis Settings* in the app:

- preserve_correlations — how strongly cross-variable relationships are retained ("none", "moderate", "high").
- coarsen_dates — logical; round dates (e.g. to month or year) so an exact event date cannot single out an individual.
- merge_rare — logical; combine infrequent category values into an "other" group to reduce re-identification risk.
- k_anon — minimum cell size for k-anonymity. Here, a quasi-identifier (QI) is a column that can identify someone only when combined with others, a cell is one shared QI combination, and suppression means blanking QI values in cells that still fall below the target. The validator allows values down to 2, but automated escape-route suggestions never pick a value below 3.
- rare_level_min_n — integer; category values seen fewer than this many times count as rare (then merged or suppressed).
- free_text_strategy — how free-text columns are treated: "categorical" (default) synthesizes them the same way as any other categorical column, with rare/near-unique values collapsed to ".other"; "drop" and "redact" are also available and are used to reinforce privacy hardening when a pre-stage check flags a free-text column as high risk.
- preserve_missingness — how closely to reproduce the original pattern of missing (NA) values ("approx", "exact", "none").

Value

An S3 object of class dataganger_spec (a named list).

Examples

```
synth_spec(purpose = "demo")
synth_spec(purpose = "development", n = 200, seed = 42)
synth_spec(purpose = "analytics", acknowledge_risk = TRUE)
```

synthesize_data *Synthesize a data double*

Description

Creates a synthetic copy of a dataset using the specified specification and engine. The internal engine supports schema-only (Level 1) and marginal (Level 2) synthesis. The optional synthpop engine is used for objectives that request moderate or high relationship preservation.

Usage

```
synthesize_data(data, spec, roles = NULL, engine = NULL)
```

Arguments

data	A data frame to synthesize from.
spec	A dataganger_spec object from synth_spec().
roles	Optional; a dataganger_roles object from detect_roles(). Informs column treatment but does not override the spec.
engine	Character or NULL. Engine to use: "auto", "internal", "marginal" (alias for "internal"), or "synthpop". When NULL, defaults to spec$engine or derives from spec$preserve_correlations.

Value

An S3 object of class dataganger_synthetic, a tibble with attributes spec, original_dims, seed_used, and generated_at.

Disabling synthpop

Set options(dataganger.disable_synthpop = TRUE) to steer auto-derived synthesis onto the internal engine even when synthpop is installed. This is intended for environments where a synthpop synthesis is undesirable or can hang unattended (for example continuous integration). An explicit engine = "synthpop" request is still honoured; only objective-derived routing is affected.

Examples

```
dat <- data.frame(x = 1:5, y = letters[1:5])
spec <- synth_spec(purpose = "demo")
syn <- synthesize_data(dat, spec)
```

`temporal_sample` *Temporal synthetic sample data*

Description

A synthetically generated dataset of 365 daily records for use as sample input in the DataGangeR Shiny app. Simulates environmental monitoring data across multiple sites. Generated with `set.seed(42)`.

Usage

```
temporal_sample
```

Format

A data frame with 365 rows and 5 columns:

date Measurement date (daily from 2023-01-01)

site_id Site identifier (SITE_A through SITE_E)

measurement Numeric measurement value (some NAs)

temperature Ambient temperature in degrees Celsius

flagged Logical quality-control flag

Source

Synthetically generated via `data-raw/temporal_sample.R`

Index

∗ **datasets**
  example_admin_claims, 7
  example_health_survey, 8
  example_registry, 9
  geographic_sample, 12
  individual_sample, 13
  temporal_sample, 24

assess_kanonymity, 2

check_code_readiness, 3
check_code_readiness(), *11*
compare_synthetic, 4
compare_synthetic(), *11*, *15*

dataganger_cli, 5
detect_roles, 5
detect_roles(), *3, 4*, *10*, *16*, *23*

enforce_kanon, 6
example_admin_claims, 7
example_health_survey, 8
example_registry, 9
export_diagnostic_package, 9
export_synthetic, 10

geographic_sample, 12

individual_sample, 13

looks_aggregated, 13

make_agent_bundle, 14

plot_comparison, 15
privacy_check, 15
privacy_check(), *11*
profile_data, 16
profile_data(), *5*, *10*, *19, 20*

read_input, 17
read_input(), *14*
readr::read_csv(), *17*
readxl::read_excel(), *17*
report_issue, 18
run_app, 19

suggest_min_rows, 19
synth_spec, 21
synth_spec(), *14*, *23*
synthesize_data, 23
synthesize_data(), *3, 4*, *11*, *21, 22*

temporal_sample, 24
tibble::tibble(), *17*