Package 'FakeDataR'

October 6, 2025

Title Privacy-Preserving Synthetic Data for 'LLM' Workflows

Version 0.2.2

Description Generate privacy-preserving synthetic datasets that mirror structure, types, factor levels, and missingness; export bundles for 'LLM' workflows (data plus 'JSON' schema and guidance); and build fake data directly from 'SQL' database tables without reading real rows. Methods are related to approaches in Nowok, Raab and Dibben (2016) <doi:10.32614/RJ-2016-019> and the foundation-model overview by Bommasani et al. (2021) <doi:10.48550/arXiv.2108.07258>.

License MIT + file LICENSE

URL https://zobaer09.github.io/FakeDataR/,
https://github.com/zobaer09/FakeDataR

BugReports https://github.com/zobaer09/FakeDataR/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, jsonlite, zip

Suggests readr, testthat (>= 3.0.0), knitr, rmarkdown, DBI, RSQLite, tibble, nycflights13, palmerpenguins, gapminder, arrow, withr

VignetteBuilder knitr, rmarkdown

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Zobaer Ahmed [aut, cre]

Maintainer Zobaer Ahmed <zunnun09@gmail.com>

Repository CRAN

Date/Publication 2025-10-06 08:10:19 UTC

Contents

`detect_sensitive_columns`

Detect sensitive columns by name

Description

Uses a broad, configurable regex library to match likely PII columns. You can extend it with `extra_patterns` (they get ORed in) or replace everything with a single `override_regex`.

Usage

```
detect_sensitive_columns(x_names, extra_patterns = NULL, override_regex = NULL)
```

Arguments

`x_names`	Character vector of column names to check.
`extra_patterns`	Character vector of additional regexes to OR in. Examples: c("MRN", "NHS", "Aadhaar", "passport")
`override_regex`	Optional single regex string that fully replaces the defaults (case-insensitive). When supplied, `extra_patterns` is ignored.

Value

Character vector of names from `x_names` that matched.

Examples

```
detect_sensitive_columns(c("id","email","home_phone","zip","notes"))
detect_sensitive_columns(names(mtcars), extra_patterns = c("^vin$", "passport"))
```

export_fake *Save a fake dataset to disk*

Description

Save a data.frame to CSV, RDS, or Parquet based on the file extension.

Usage

```
export_fake(x, path)
```

Arguments

x	A data.frame (e.g., output of `generate_fake_data()`).
path	File path. Supported extensions: `.csv`, `.rds`, `.parquet`.

Value

(Invisibly) the path written.

generate_fake_data *Generate Fake Data from Real Dataset Structure*

Description

Generate Fake Data from Real Dataset Structure

Usage

```
generate_fake_data(
  data,
  n = 30,
  category_mode = c("preserve", "generic", "custom"),
  numeric_mode = c("range", "distribution"),
  column_mode = c("keep", "generic", "custom"),
  custom_levels = NULL,
  custom_names = NULL,
  seed = NULL,
  verbose = FALSE,
  sensitive = NULL,
  sensitive_detect = TRUE,
  sensitive_strategy = c("fake", "drop"),
  normalize = TRUE
)
```

Arguments

`data`	A tabular object; will be coerced via `prepare_input_data()`.
`n`	Rows to generate (default 30).
`category_mode`	One of "preserve","generic","custom". • preserve: sample observed categories by empirical frequency (keeps factors) • generic: replace categories with "Category A/B/..." • custom: use `custom_levels[[colname]]` if provided
`numeric_mode`	One of "range","distribution". • range: uniform between min/max (integers stay integer-like) • distribution: sample observed values with replacement
`column_mode`	One of "keep","generic","custom". • keep: keep original column names `var1..varP` (mapping in `attr(name_map)`) • custom: use `custom_names` named vector (old -> new)
`custom_levels`	optional named list of allowed levels per column (for
`custom_names`	optional named character vector old->new (for `column_mode="custom"`).
`seed`	Optional RNG seed.
`verbose`	Logical; print progress.
`sensitive`	Optional character vector of original column names to treat as sensitive.
`sensitive_detect`	Logical; auto-detect common sensitive columns by name.
`sensitive_strategy`	One of "fake","drop". Only applied if any sensitive columns exist.
`normalize`	Logical; lightly normalize inputs (trim, %→numeric, short date-times→POSIXct).

Value

A data.frame of n rows with attributes:

- `name_map` (named chr: original -> output)
- `column_mode` (chr)
- `sensitive_columns` (chr; original names)
- `dropped_columns` (chr; original names that were dropped)

generate_fake_from_schema

Generate fake data from a DB schema data.frame

Description

Generate fake data from a DB schema data.frame

Usage

```
generate_fake_from_schema(sch_df, n = 30, seed = NULL)
```

Arguments

sch_df	A data.frame returned by schema_from_db().
n	Number of rows to generate.
seed	Optional integer seed for reproducibility.

Value

A base data.frame with n rows and one column per schema entry. Column classes follow the schema type values (integer, numeric, character, logical, Date, POSIXct); missingness is injected when nullable is TRUE.

generate_fake_posixct_column

Generate a Fake POSIXct Column

Description

Create synthetic timestamps either by mimicking an existing POSIXct vector (using its range and NA rate) or by sampling uniformly between start and end.

Usage

```
generate_fake_posixct_column(
  like = NULL,
  n = NULL,
  start = NULL,
  end = NULL,
  tz = "UTC",
  na_prop = NULL
)
```

Arguments

like	Optional POSIXct vector to mimic. If supplied, n defaults to length(like), the output range matches range(like, na.rm = TRUE), and the NA rate is copied unless you override with na_prop.
n	Number of rows to generate. Required when like is NULL.
start, end	Optional POSIXct bounds to sample between when like is NULL.
tz	Timezone to use if like has no tzone (default "UTC").
na_prop	Optional NA proportion to enforce in the output (0–1). If NULL and like is provided, it copies the NA rate from like. If like is NULL, defaults to 0.

Value

A POSIXct vector of length n.

---

generate_fake_with_privacy

Generate fake data with privacy controls

Description

Generates a synthetic copy of data, then optionally detects/handles sensitive columns by name. Detection uses the ORIGINAL column names and maps to output via attr(fake, "name_map") if present.

Usage

```
generate_fake_with_privacy(
  data,
  n = 30,
  level = c("low", "medium", "high"),
  seed = NULL,
  sensitive = NULL,
  sensitive_detect = TRUE,
  sensitive_strategy = c("fake", "drop"),
  normalize = TRUE,
  sensitive_patterns = NULL,
  sensitive_regex = NULL
)
```

Arguments

data	A data.frame (or coercible) to mirror.
n	Rows to generate (default same as input if NULL).
level	One of "low","medium","high".
seed	Optional RNG seed.

`sensitive` Character vector of original column names to treat as sensitive.

`sensitive_detect`
Logical; auto-detect common sensitive columns by name.

`sensitive_strategy`
One of "fake" or "drop".

`normalize` Logical; lightly normalize inputs.

`sensitive_patterns`
Optional named list of patterns to treat as sensitive (e.g., list(id = "...", email = "...", phone = "...")). Overrides defaults.

`sensitive_regex`
Optional fully-combined regex (single string) to detect sensitive columns by name. If supplied, it is used instead of defaults.

Details

Generate fake data with privacy controls

Value

data.frame with attributes: sensitive_columns, dropped_columns, name_map

---

`generate_llm_prompt` *Create a copy-paste prompt for LLMs*

Description

Create a copy-paste prompt for LLMs

Usage

```
generate_llm_prompt(
  fake_path,
  schema_path = NULL,
  notes = NULL,
  write_file = TRUE,
  path = dirname(fake_path),
  filename = "README_FOR_LLM.txt"
)
```

Arguments

`fake_path` Path to the fake data file (CSV/RDS/Parquet).

`schema_path` Optional path to the JSON schema.

`notes` Optional extra notes to append for the analyst/LLM.

`write_file` Write a README txt next to the files? Default TRUE.

`path` Output directory for the README if write_file = TRUE.

`filename` README file name. Default "README_FOR_LLM.txt".

Value

The prompt string (invisibly returns the file path if written).

`llm_bundle` *Create a fake-data bundle for LLM workflows*

Description

Generates fake data, writes files (CSV/RDS/Parquet), writes a scrubbed JSON schema, and optionally writes a README prompt and a single ZIP file containing everything.

Usage

```
llm_bundle(
  data,
  n = 30,
  level = c("medium", "low", "high"),
  formats = c("csv", "rds"),
  path = tempdir(),
  filename = "fake_bundle",
  seed = NULL,
  write_prompt = TRUE,
  zip = FALSE,
  prompt_filename = "README_FOR_LLM.txt",
  zip_filename = NULL,
  sensitive = NULL,
  sensitive_detect = TRUE,
  sensitive_strategy = c("fake", "drop"),
  normalize = FALSE
)
```

Arguments

`data`	A data.frame (or coercible) to mirror.
`n`	Number of rows in the fake dataset (default 30).
`level`	Privacy level: "low", "medium", or "high". Controls stricter defaults.
`formats`	Which data files to write: any of "csv","rds","parquet".
`path`	Folder to write outputs. Default: `tempdir()`.
`filename`	Base file name (no extension). Example: "demo_bundle". This becomes files like "demo_bundle.csv", "demo_bundle.rds", etc.
`seed`	Optional RNG seed for reproducibility.
`write_prompt`	Write a README_FOR_LLM.txt next to the data? Default TRUE.
`zip`	Create a single zip archive containing data + schema + README? Default FALSE.

prompt_filename
: Name for the README file. Default "README_FOR_LLM.txt".

zip_filename
: Optional custom name for the ZIP file (no path). If NULL (default), it is derived as paste0(filename, ".zip"), e.g. "demo_bundle.zip".

sensitive
: Character vector of column names to treat as sensitive (optional).

sensitive_detect
: Logical, auto-detect common sensitive columns (id/email/phone). Default TRUE.

sensitive_strategy
: "fake" (replace with realistic fakes) or "drop". Default "fake".

normalize
: Logical; if TRUE, attempt light auto-normalization before faking.

Details

Tips Avoid using angle brackets in examples; prefer plain tokens like NAME or FILE_NAME. If you truly want bracket glyphs, use Unicode ⟨name⟩ ⟩name⟨.

Value

List with paths: $data_paths (named), $schema_path, $readme_path (optional), $zip_path (optional), and $fake (data.frame).

---

llm_bundle_from_db *Build an LLM bundle directly from a database table*

Description

Reads just the schema from table on conn, synthesizes n fake rows, writes a schema JSON, fake dataset(s), and a README prompt, and optionally zips them into a single archive.

Usage

```
llm_bundle_from_db(
  conn,
  table,
  n = 30,
  level = c("medium", "low", "high"),
  formats = c("csv", "rds"),
  path = tempdir(),
  filename = "fake_from_db",
  seed = NULL,
  write_prompt = TRUE,
  zip = FALSE,
  zip_filename = NULL,
  sensitive_strategy = c("fake", "drop")
)
```

Arguments

conn	A DBI connection.
table	Character scalar: table name to read.
n	Number of rows in the fake dataset (default 30).
level	Privacy level: "low", "medium", or "high". Controls stricter defaults.
formats	Which data files to write: any of "csv","rds","parquet".
path	Folder to write outputs. Default: `tempdir()`.
filename	Base file name (no extension). Example: "demo_bundle". This becomes files like "demo_bundle.csv", "demo_bundle.rds", etc.
seed	Optional RNG seed for reproducibility.
write_prompt	Write a README_FOR_LLM.txt next to the data? Default TRUE.
zip	Create a single zip archive containing data + schema + README? Default FALSE.
zip_filename	Optional custom name for the ZIP file (no path). If `NULL` (default), it is derived as `paste0(filename, ".zip")`, e.g. `"demo_bundle.zip"`.
sensitive_strategy	"fake" (replace with realistic fakes) or "drop". Default "fake".

Value

Invisibly, a list with useful paths:

- `schema_path` – schema JSON
- `files` – vector of written fake-data files
- `zip_path` – zip archive path (if `zip = TRUE`)

Examples

```
if (requireNamespace("DBI", quietly = TRUE) &&
    requireNamespace("RSQLite", quietly = TRUE)) {
  con <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
  on.exit(DBI::dbDisconnect(con), add = TRUE)
  DBI::dbWriteTable(con, "cars", head(cars, 20), overwrite = TRUE)
  out <- llm_bundle_from_db(
    con, "cars",
    n = 100, level = "medium",
    formats = c("csv","rds"),
    path = tempdir(), filename = "db_bundle",
    seed = 1, write_prompt = TRUE, zip = TRUE
  )
}
```

`prepare_input_data` *Prepare Input Data: Coerce to data.frame and (optionally) normalize values*

Description

Converts common tabular objects to a base `data.frame`, and if `normalize = TRUE` it applies light, conservative value normalization:

- Converts common date/time strings to POSIXct (best-effort across several formats)
- Converts percent-like character columns (e.g. "85%") to numeric (85)
- Maps a configurable set of "NA-like" strings to `NA`, while *keeping* common survey responses like "not applicable" or "prefer not to answer" as **real levels**
- Normalizes yes/no character columns to an ordered factor `c("no","yes")`

Usage

```
prepare_input_data(
  data,
  normalize = TRUE,
  na_strings = c("", "NA", "N/A", "na", "No data", "no data"),
  keep_as_levels = c("not applicable", "prefer not to answer", "unsure"),
  percent_detect_threshold = 0.6,
  datetime_formats = c("%m/%d/%Y %H:%M:%S", "%m/%d/%Y %H:%M",
    "%Y-%m-%d %H:%M:%S", "%Y-%m-%d %H:%M", "%Y-%m-%dT%H:%M:%S",
    "%Y-%m-%dT%H:%M", "%m/%d/%Y", "%Y-%m-%d")
)
```

Arguments

`data`	An object coercible to `data.frame` (data.frame/**tibble**/data.table/matrix/list, etc.)
`normalize`	Logical, run value normalization step (default `TRUE`).
`na_strings`	Character vector that should become `NA` (default: `c("", "NA", "N/A", "na", "No data", "no data")`).
`keep_as_levels`	Character vector that should be **kept as values** (not `NA`), e.g., survey choices (default: `c("not applicable", "prefer not to answer", "unsure")`). Matching is case-insensitive.
`percent_detect_threshold`	Proportion of non-missing values that must contain `%` before converting a character column to numeric (default `0.6`).
`datetime_formats`	Candidate formats tried (in order) when parsing date-times strings. The best-fitting format (most successful parses) is used. Defaults cover `mm/dd/yyyy HH:MM(:SS)?`, ISO-8601, and date-only.

Value

A base `data.frame`.

`schema_from_db` *Extract a table schema from a DB connection*

Description

Returns a data frame describing the columns of a database table.

Usage

```
schema_from_db(conn, table, level = c("medium", "low", "high"))
```

Arguments

`conn`	A DBI connection.
`table`	Character scalar: table name to introspect.
`level`	Privacy preset to annotate in schema metadata: one of "low", "medium", "high". Default "medium".

Value

A data.frame with column metadata (e.g., name, type).

Examples

```
if (requireNamespace("DBI", quietly = TRUE) &&
    requireNamespace("RSQLite", quietly = TRUE)) {
  con <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
  on.exit(DBI::dbDisconnect(con), add = TRUE)
  DBI::dbWriteTable(con, "mtcars", mtcars[1:3, ])
  sc <- schema_from_db(con, "mtcars")
  head(sc)
}
```

validate_fake *Validate a fake dataset against the original*

Description

Compares classes, NA/blank proportions, and simple numeric ranges.

Usage

```
validate_fake(original, fake, tol = 0.15)
```

Arguments

original	data.frame
fake	data.frame (same columns)
tol	numeric tolerance for proportion differences (default 0.15)

Value

data.frame summary by column

zip_llm_bundle *Zip a set of files for easy sharing*

Description

Zip a set of files for easy sharing

Usage

```
zip_llm_bundle(files, zipfile)
```

Arguments

files	Character vector of file paths.
zipfile	Path to the zip file to create.

Value

The path to the created zip file.

Index