Package: synadam (via r-universe)

July 23, 2026

Title Generate Synthetic ADaM Datasets

Version 0.3.3

Description Generates synthetic ADaM (Analysis Data Model) datasets from real clinical trial data. Preserves the structure of real datasets (column names, value ranges, relationships between treatment and flag columns) while removing identifiable patient information. Supports subject-level (ADSL), longitudinal (BDS), occurrence (OCCDS), and time-to-event (TTE) dataset types.

License MIT + file LICENSE

URL https://github.com/Novartis/synadam

BugReports https://github.com/Novartis/synadam/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports checkmate, dplyr, glue, haven, rlang, stats, tidyr, tools, utils, yaml

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

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

Date/Publication 2026-07-16 12:33:14 UTC

RemoteUrl https://github.com/novartis/synadam

RemoteRef HEAD

RemoteSha 21c6f19522b69155801912952e9fc483aa6db897

Contents

`generate_study_config` *Generate a draft study config YAML from a directory of ADaM files.*

Description

Scans a directory of `.sas7bdat` files, infers `dataset_type` and per-dataset glimpse arguments using CDISC filename and column-name conventions, and writes a draft YAML config suitable for `simulate_study()`. Inferred fields are tagged with inline `# REVIEW` comments where the signal is weak so users know what to verify before running the simulation.

Usage

```
generate_study_config(
  adam_dir,
  output_dir,
  config_yaml_name = "synadam_config.yaml",
  seed = 123
)
```

Arguments

adam_dir	character(1) - directory containing .sas7bdat files.
output_dir	character(1) - directory that both the generated config YAML and the synthetic .rds outputs are written to. Created recursively if it does not exist. This same path is recorded as the top-level output_dir field in the YAML, so simulate_study() writes its syn_*.rds files alongside the config.
config_yaml_name	character(1) - filename for the generated config within output_dir. Defaults to "synadam_config.yaml". If the file already exists, it is silently overwritten.
seed	integer(1) / numeric(1) or NULL - value to write for the top-level seed field in the YAML. Defaults to 123. If NULL, a "REVIEW: ..." placeholder is written instead.

Details

Only files whose basename matches ad*.sas7bdat (case-insensitive) are considered. Any other files in adam_dir are skipped with a single warning listing them.

Value

character(1) - the path the YAML was written to (file.path(output_dir, config_yaml_name)).

Detection rules

Filename match (case-insensitive, basename without extension):

- starts with adsl -> adsl
- starts with adae/adcm/admh/addv -> occds
- starts with adtte -> tte
- any other name (including ad*) -> fall through to column inspection

Column inspection rules (checked in order; USUBJID is required in every ADaM dataset, so it is not used as a discriminator). ADSL is classified from the filename only - any non-adsl* file that lacks BDS/TTE/OCCDS column signatures falls through to "REVIEW":

- has PARAM -> bds (checked before CNSR so oncology hybrid datasets like ADRECIST that carry TTE-derived parameters inline classify as the longitudinal BDS they actually are)
- has CNSR -> tte (CNSR is the only structurally required TTE variable in ADaM; EVNTDESC is conventional but not universal)
- has any *SEQ column -> occds
- otherwise -> "REVIEW" placeholder + warning

Errors if no ADSL file is detected; simulate_study() requires exactly one. If multiple ADSL candidates are detected, a file named exactly adsl.sas7bdat (case-insensitive) is preferred; otherwise the alphabetically-first candidate is used. The picked file is kept and the rest are dropped with a warning.

Column-role detection

Within a classified dataset, the column-role inference picks up:

- `id_cols`: `USUBJID` and `SUBJID` when present.
- `treatment_cols` (ADSL): `ACTARM`, `ACTARMCD`, `ARM`, `ARMCD`, `TRTSEQP`, `TRTSEQA` and any `TRT##A/TRT##AN/TRT##AL/TRT#` matches, plus `TRT##(A|P)GR\d*` and `TRT##(A|P)GR\d*N` group variants.
- `visit_cols` (BDS): `AVISIT`, `AVISITN`, `VISIT`, `VISITNUM`, `ATPT`, `ATPTN` when present.
- `param_cols` (BDS, TTE): `PARAM`, `PARAMCD`, and any `PARCAT*` columns.
- `seq_col` (OCCDS): the first known domain-coded `*SEQ` column (`AESEQ`, `CMSEQ`, `MHSEQ`, `PRSEQ`, `EXSEQ`, `DSSEQ`); falls back to the first `*SEQ` column if none of those are present.
- `censor_cols` (TTE): `CNSR`, `EVNTDESC`, `CNSDTDSC` - only emitted when two or more of these are detected. A single `CNSR` flows through as a regular column and `censor_cols` is omitted.
- `ordered_col_sets`: X/XN/XL groups where two or three variants of a base column name co-exist.

What is NOT auto-inferred

Some config decisions are opinionated and intended to be pruned or extended by the user after generation:

- `flag_cols` emits every `*FL` column found. The user is expected to trim this list down to the flags they want to preserve.
- Non-FL flag-like indicators (e.g. `AESER`) are not detected.
- Ordered column sets that share semantics but not naming (e.g. `AEBODSYS/AEDECOD`) are not auto-paired; add them by hand.
- For BDS and TTE datasets `param_cols` auto-includes `PARAM`, `PARAMCD`, and any `PARCAT*` columns present, but NOT `PARAMTYP` or `PARAMN`; add these manually if relevant.
- For BDS, OCCDS, and TTE datasets `ordered_col_sets` excludes any column also present in the chosen ADSL. ADSL-shared columns are pulled in via the ADSL spine at simulation time, so listing them here would duplicate the column.

Examples

```
yaml_path <- generate_study_config(
  adam_dir,
  output_dir = file.path(tempdir(), "syn_out")
)
yaml_path
```

`glimpse` *Glimpse a vector.*

Description

Extracts a statistical summary from a vector that preserves its structure while enabling synthetic data generation. The summary type varies by vector class: character vectors output unique values, numeric vectors output min/max values, and date/time vectors output earliest/latest dates.

Usage

```
glimpse(x, ...)

## S3 method for class 'character'
glimpse(x, col_name, na_mode = "mirror", seed = NULL, ...)

## S3 method for class 'Date'
glimpse(x, col_name, na_mode = "mirror", seed = NULL, ...)

## S3 method for class 'POSIXct'
glimpse(x, col_name, na_mode = "mirror", seed = NULL, ...)

## S3 method for class 'difftime'
glimpse(x, col_name, na_mode = "mirror", seed = NULL, ...)

## S3 method for class 'numeric'
glimpse(x, col_name, na_mode = "mirror", seed = NULL, ...)
```

Arguments

`x`	`vector` - input vector.
`...`	additional arguments passed to methods.
`col_name`	`character` - column name corresponding to the input vector.
`na_mode`	`character` - "mirror" to capture NA positions, "none" to ignore NAs in simulation.
`seed`	`integer` - random seed for reproducibility.

Value

`summary` - contains summary of the input vector.

Methods (by class)

- `glimpse(character)`: glimpse a character vector using `.get_unique_values()`.
- `glimpse(Date)`: glimpse a Date vector into a summary containing the min, max, and NA positions.

- `glimpse(POSIXct)`: glimpse a POSIXct vector into a summary containing the min, max, and NA positions.
- `glimpse(difftime)`: glimpse a difftime vector into a summary containing the min, max, units, and NA positions.
- `glimpse(numeric)`: glimpse a numeric vector into a summary containing the min, max, and NA positions. Detects if values are integer or double.

Examples

```
# Glimpse a character vector, then simulate synthetic values from it.
summary <- glimpse(c("A", "B", "B", "C"), col_name = "column_name")
simulate(summary, output_length = 10, seed = 1)
```

`glimpse_adsl` *Glimpse an ADSL dataset.*

Description

Summarizes an ADSL (subject-level) dataset for synthetic simulation by preserving treatment-flag relationships, capturing ordered column combinations and extracting summaries for all other variables.

Usage

```
glimpse_adsl(
  adsl,
  id_cols,
  treatment_cols,
  flag_cols = character(),
  ordered_col_sets = list(),
  seed = NULL
)
```

Arguments

`adsl`	`data.frame` - input ADSL dataset.
`id_cols`	`character` - ID columns e.g. "USUBJID".
`treatment_cols`	`character` - treatment columns e.g. "TRT01A".
`flag_cols`	`character` - flag columns e.g. "SAFFL". Default is an empty character vector (no flag columns).
`ordered_col_sets`	`list<character>` - each element is a character vector naming columns whose combinations should be preserved e.g. `list(c("REGION1", "REGION1N"))`. Default is an empty list.
`seed`	`integer` - random seed for reproducibility.

Value

summary_adsl - list of summary objects, with each summarising a column or set of columns in the input ADSL.

Examples

```
# Glimpse the ADSL, then simulate a synthetic version from the summary.
adsl_summary <- glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN"),
  flag_cols = c("SAFFL", "ITTFL", "EFFFL"),
  ordered_col_sets = list(c("REGION1", "REGION1N")),
  seed = 42
)
syn_adsl <- simulate_adsl(adsl_summary, seed = 42)
head(syn_adsl)
```

---

glimpse_bds *Glimpse a BDS dataset.*

Description

Summarizes a BDS (Basic Data Structure) e.g. ADLB, ADVS dataset for synthetic simulation by extracting per-subject parameter/visit profiles, preserving relationships with ADSL columns, and extracting summaries of all other variables. All PARAMCDs in the dataset are included.

Usage

```
glimpse_bds(
  bds,
  syn_adsl,
  id_cols,
  param_cols,
  visit_cols = character(),
  flag_cols = character(),
  ordered_col_sets = list(),
  seed = NULL
)
```

Arguments

bds	data.frame - input BDS dataset e.g. ADLB.
syn_adsl	data.frame - synthetic ADSL to obtain subject-level data from. Should be created by simulate_adsl().
id_cols	character - ID column names that identify subjects e.g. c("USUBJID"). These columns must exist in syn_adsl and will be taken from there (never simulated).

param_cols	character - parameter columns e.g. c("PARAM", "PARAMCD").
visit_cols	character - visit columns e.g. c("AVISIT", "AVISITN"). Default is an empty character vector (no visit columns).
flag_cols	character - flag columns that will be sampled as Y/N in proportion to input e.g. c("ANL01FL"). Default is an empty character vector (no flag columns).
ordered_col_sets	list<character> - each element is a character vector naming columns whose combinations should be preserved e.g. list(c("REGION1", "REGION1N")). Default is an empty list.
seed	integer - random seed for reproducibility.

Value

summary_bds - summary object for BDS simulation.

Examples

```
# BDS simulation needs a synthetic ADSL first to provide the subject spine.
syn_adsl <- simulate_adsl(glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN")
))

bds_summary <- glimpse_bds(
  adlb,
  syn_adsl,
  id_cols = "USUBJID",
  param_cols = c("PARAM", "PARAMCD"),
  visit_cols = c("AVISIT", "AVISITN"),
  flag_cols = "ANL01FL"
)
syn_adlb <- simulate_bds(bds_summary, seed = 42)
head(syn_adlb)
```

---

glimpse_id *Glimpse an ID vector.*

Description

Glimpse an ID vector.

Usage

```
glimpse_id(x, col_name)
```

Arguments

x	vector - contains ID values.
col_name	character - column name corresponding to the input vector.

Value

summary - contains column name for ID generation.

Examples

```
# Glimpse an ID column, then simulate fresh sequential IDs.
summary <- glimpse_id(c("S1", "S2", "S3"), col_name = "USUBJID")
simulate(summary, output_length = 5)
```

---

glimpse_occds *Glimpse an OCCDS dataset.*

Description

Summarizes an OCCDS (Occurrence Data Structure) dataset for simulation. Works with ADAE, ADCM, ADMH, ADDV and other occurrence-based datasets.

Usage

```
glimpse_occds(
  occds,
  syn_adsl,
  id_cols,
  seq_col,
  flag_cols = character(0),
  ordered_col_sets = list(),
  seed = NULL
)
```

Arguments

occds	data.frame - input OCCDS dataset e.g. ADAE, ADCM.
syn_adsl	data.frame - synthetic ADSL to obtain subject-level data from. Should be created by simulate_adsl().
id_cols	character - ID column names that identify subjects e.g. c("USUBJID"). These columns must exist in syn_adsl and will be taken from there (never simulated).
seq_col	character - sequence column name e.g. "AESEQ", "CMSEQ".
flag_cols	character - flag columns that will be sampled as Y/N in proportion to input e.g. c("ANL01FL"). Default is an empty character vector (no flag columns).

ordered_col_sets
: list<character> - each element is a character vector naming columns whose combinations should be preserved e.g. list(c("REGION1", "REGION1N")). Default is an empty list.

seed
: integer - random seed for reproducibility.

Value

summary_occds - summary object for OCCDS simulation.

Examples

```
# OCCDS simulation needs a synthetic ADSL first to provide the subject spine.
syn_adsl <- simulate_adsl(glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN")
))

occds_summary <- glimpse_occds(
  adae,
  syn_adsl,
  id_cols = "USUBJID",
  seq_col = "AESEQ",
  ordered_col_sets = list(c("AEBODSYS", "AEDECOD"))
)
syn_adae <- simulate_occds(occds_summary, seed = 42)
head(syn_adae)
```

---

glimpse_occurrence_id_seq

Glimpse occurrence counts, ID columns, and sequence column.

Description

Counts the number of occurrences (records) per subject in an OCCDS dataset and captures the ID and sequence column names needed for spine construction.

Usage

```
glimpse_occurrence_id_seq(df, id_cols, seq_col)
```

Arguments

df
: data.frame - OCCDS dataset containing ID and sequence columns.

id_cols
: character - name(s) of ID column(s) to group by.

seq_col
: character - name of the sequence column.

Value

summary_occurrence_id_seq - contains occurrence counts per subject, ID column names, and sequence column name.

Examples

```
# Capture per-subject occurrence counts (used within glimpse_occds()).
glimpse_occurrence_id_seq(adae, id_cols = "USUBJID", seq_col = "AESEQ")
```

---

glimpse_ordered *Glimpse ordered columns.*

Description

Ordered columns are those which should be simulated together, such as PARAM/PARAMCD or REGION1/REGION1N/REGION2/REGION2N. Rare combinations (count = 1) are masked for privacy protection.

Usage

```
glimpse_ordered(df)
```

Arguments

df	data.frame - contains ordered columns.

Value

summary - contains column names, unique combinations, and NA positions.

Examples

```
df <- data.frame(
  REGION1 = c("North America", "North America", "Europe", "Europe"),
  REGION1N = c(1, 1, 2, 2)
)
summary <- glimpse_ordered(df)
simulate(summary, output_length = 6, seed = 1)
```

glimpse_param_visits *Glimpse parameter and visit columns.*

Description

Extracts per-subject param/visit profiles from a BDS dataset. Rare param/visit combinations (count = 1) are masked for privacy protection. Subjects are grouped by their set of surviving param/visit combinations to form profiles, which are used by `simulate_bds()` to construct the simulation spine.

Usage

```
glimpse_param_visits(df, id_cols, param_visit_cols)
```

Arguments

`df`	`data.frame` - contains ID and parameter/visit columns.
`id_cols`	`character` - ID column(s) to identify subjects e.g. c("USUBJID").
`param_visit_cols`	`character` - parameter and visit column names e.g. c("PARAM", "PARAMCD", "AVISIT", "AVISITN").

Value

`summary_param_visits` - contains deduplicated profiles with masked counts and the unique param/visit combinations (after masking).

Examples

```
# Build per-subject parameter/visit profiles (used within glimpse_bds()).
glimpse_param_visits(
  adlb,
  id_cols = "USUBJID",
  param_visit_cols = c("PARAM", "PARAMCD", "AVISIT", "AVISITN")
)
```

glimpse_params *Glimpse parameter columns.*

Description

Extracts unique combinations of parameter columns without filtering. Used for TTE datasets where all PARAMCDs are simulated. Rare combinations (count = 1) are masked for privacy protection.

Usage

```
glimpse_params(df)
```

Arguments

df	data.frame - contains parameter columns.

Value

summary_params - contains unique parameter combinations.

Examples

```
df <- data.frame(
  PARAM = c("Overall Survival", "Overall Survival", "Progression Free"),
  PARAMCD = c("OS", "OS", "PFS")
)
glimpse_params(df)
```

---

glimpse_study *Glimpse all ADaM datasets for a study.*

Description

Reads the YAML configuration used by simulate_study(), runs the appropriate glimpse_*() for each dataset, and saves the collected summaries (plus seed and version metadata) to a single .rds file.

Usage

```
glimpse_study(config_path, study_summary_path)
```

Arguments

config_path	character - path to YAML configuration file. Has the same structure as the one consumed by simulate_study().
study_summary_path	character - path to write the study summary .rds file. The parent directory will be created if it does not exist.

Details

The resulting study summary can later be passed to simulate_study_from_summary() to generate the synthetic datasets, decoupling the glimpse phase (which requires access to real .sas7bdat files) from the simulate phase.

Value

NULL (invisibly). The study summary is a named list with elements summaries (named list of glimpse summary objects, keyed by dataset name from the YAML), seed, synadam_version, and glimpsed_at.

See Also

simulate_study_from_summary(), simulate_study().

Examples

```
# Generate a config for the staged ADaM datasets.
yaml_path <- generate_study_config(
  adam_dir,
  output_dir = file.path(tempdir(), "syn_glimpse_out")
)

# Glimpse phase: write the study summary (decoupled from the SAS files).
summary_path <- tempfile(fileext = ".rds")
glimpse_study(yaml_path, summary_path)

# Simulate phase: generate synthetic datasets from the study summary.
out_dir <- file.path(tempdir(), "syn_glimpse_out")
simulate_study_from_summary(summary_path, out_dir)
list.files(out_dir, pattern = "\\.rds$")
```

---

glimpse_treatment_flag

Glimpse treatment and flag columns together.

Description

Captures unique combinations of treatment and flag columns with their counts, preserving the relationship between treatments and analysis flags. Combinations with count = 1 are masked for privacy protection by adding their counts to the most common combination.

Usage

```
glimpse_treatment_flag(df)
```

Arguments

df	data.frame - contains treatment and flag columns.

Value

summary - contains treatment-flag combinations with counts.

Examples

```
df <- data.frame(
  TRT01A = c("Placebo", "Placebo", "Drug A", "Drug A"),
  SAFFL = c("Y", "Y", "Y", "N")
)
summary <- glimpse_treatment_flag(df)
# simulate() preserves the exact treatment-flag counts (no output_length).
simulate(summary)
```

`glimpse_tte` *Glimpse a TTE dataset.*

Description

Summarizes a TTE (Time-to-Event) dataset for synthetic simulation by extracting parameter combinations, preserving censoring column relationships, and extracting summaries of all other variables.

Usage

```
glimpse_tte(
  tte,
  syn_adsl,
  param_cols,
  censor_cols = character(),
  flag_cols = character(),
  ordered_col_sets = list(),
  seed = NULL
)
```

Arguments

`tte`	`data.frame` - input TTE dataset e.g. ADTTE.
`syn_adsl`	`data.frame` - synthetic ADSL to obtain subject-level data from. Should be created by `simulate_adsl()`.
`param_cols`	`character` - parameter columns e.g. c("PARAM", "PARAMCD").
`censor_cols`	`character` - censoring columns e.g. c("CNSR", "EVNTDESC", "CNSDTDSC"). These columns are treated as an ordered set to preserve co-occurrence patterns. Only needed when multiple censor columns should be kept together. Default is an empty character vector (censor columns are treated as regular columns).
`flag_cols`	`character` - flag columns that will be sampled as Y/N in proportion to input e.g. c("ANL01FL"). Default is an empty character vector (no flag columns).
`ordered_col_sets`	`list<character>` - each element is a character vector naming columns whose combinations should be preserved e.g. `list(c("REGION1", "REGION1N"))`. Default is an empty list.
`seed`	`integer` - random seed for reproducibility.

Value

summary_tte - summary object for TTE simulation.

Examples

```
# TTE simulation needs a synthetic ADSL to provide the subject spine.
syn_adsl <- data.frame(
  USUBJID = as.character(1:5),
  TRT01A = c("Placebo", "Placebo", "Drug A", "Drug A", "Drug A"),
  TRT01AN = c(1, 1, 2, 2, 2)
)
tte <- data.frame(
  USUBJID = as.character(rep(1:3, each = 2)),
  PARAM = rep(c("Overall Survival", "Progression Free Survival"), 3),
  PARAMCD = rep(c("OS", "PFS"), 3),
  CNSR = c(0, 1, 0, 1, 1, 0),
  EVNTDESC = c(
    "DEATH", "COMPLETED", "DEATH", "COMPLETED", "COMPLETED", "PROGRESSION"
  ),
  AVAL = c(365, 500, 180, 250, 600, 120)
)

tte_summary <- glimpse_tte(
  tte,
  syn_adsl,
  param_cols = c("PARAM", "PARAMCD"),
  censor_cols = c("CNSR", "EVNTDESC")
)
syn_tte <- simulate_tte(tte_summary, seed = 42)
head(syn_tte)
```

---

read_sas7bdat *Read in an ADaM dataset.*

Description

Loads in a ADaM dataset and and convert all missing values to NA.

Usage

```
read_sas7bdat(adam_path)
```

Arguments

adam_path	character - path to the .sas7bdat file containing the ADaM dataset.

Value

data.frame - ADaM dataset.

Examples

```
adam <- read_sas7bdat(sas_path)
head(adam)
```

`simulate.summary_ordered` *Simulate a vector from a summary object.*

Description

Simulate a vector from a summary object.

Usage

```
## S3 method for class 'summary_ordered'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_treatment_flag'
simulate(summary, ...)

simulate(summary, ...)

## S3 method for class 'summary_character'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_Date'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_POSIXct'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_difftime'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_integer'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_double'
simulate(summary, output_length, seed = NULL, ...)

## S3 method for class 'summary_id'
simulate(summary, output_length, ...)

## S3 method for class 'summary_flag'
simulate(summary, output_length, seed = NULL, ...)
```

Arguments

summary	summary - output from a glimpse function.
output_length	integer - number of rows to simulate.
seed	integer - random seed for reproducibility.
...	additional arguments passed to methods.

Value

data.frame - the simulated dataset.

Methods (by class)

- simulate(summary_ordered): simulate ordered columns by sampling from unique combinations.
- simulate(summary_treatment_flag): simulate treatment and flag columns by preserving exact counts for each combination.
- simulate(summary_character): simulate a character vector by sampling from unique values.
- simulate(summary_Date): simulate a Date vector by sampling uniformly between min and max.
- simulate(summary_POSIXct): simulate a POSIXct vector by sampling uniformly between min and max.
- simulate(summary_difftime): simulate a difftime vector by sampling uniformly between min and max.
- simulate(summary_integer): simulate an integerish vector by sampling uniformly between min and max and rounding.
- simulate(summary_double): simulate a double vector by sampling uniformly between min and max.
- simulate(summary_id): simulate an ID vector by generating sequential IDs.
- simulate(summary_flag): simulate a flag vector by sampling Y/N in proportion to input.

Examples

```
# Summarise a vector with glimpse().
summary <- glimpse(c("A", "B", "B", "C"), col_name = "column_name")
simulate(summary, output_length = 10, seed = 1)
```

simulate_adsl *Simulate a synthetic ADSL.*

Description

Generates synthetic subject-level data from an ADSL summary, preserving the structure of the original dataset.

Usage

```
simulate_adsl(adsl_summary, seed = NULL)
```

Arguments

adsl_summary	summary_adsl - contains summary objects that describe an ADSL dataset. Created by glimpse_adsl().
seed	integer - random seed for reproducibility.

Value

data.frame - synthetic ADSL with the same columns (and column order) as the dataset that was glimpsed, and one row per subject.

Examples

```
adsl_summary <- glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN"),
  flag_cols = "SAFFL"
)
syn_adsl <- simulate_adsl(adsl_summary, seed = 42)
head(syn_adsl)
```

simulate_bds *Simulate a synthetic BDS dataset.*

Description

Generates synthetic BDS data by creating a spine from parameter/visit combinations and ADSL subject records, then populating remaining columns, preserving the original data structure.

Usage

```
simulate_bds(bds_summary, seed = NULL)
```

Arguments

bds_summary	summary_bds - summary object created by glimpse_bds().
seed	integer - random seed for reproducibility.

Value

data.frame - synthetic BDS dataset.

Examples

```
syn_adsl <- simulate_adsl(glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN")
))
bds_summary <- glimpse_bds(
  adlb,
  syn_adsl,
  id_cols = "USUBJID",
  param_cols = c("PARAM", "PARAMCD"),
  visit_cols = c("AVISIT", "AVISITN")
)
syn_adlb <- simulate_bds(bds_summary, seed = 42)
head(syn_adlb)
```

---

simulate_occds *Simulate a synthetic OCCDS dataset.*

Description

Generates synthetic OCCDS data from a summary created by glimpse_occds().

Usage

```
simulate_occds(occds_summary, seed = NULL)
```

Arguments

occds_summary	summary_occds - summary object created by glimpse_occds().
seed	integer - random seed for reproducibility.

Value

data.frame - synthetic OCCDS dataset.

Examples

```
syn_adsl <- simulate_adsl(glimpse_adsl(
  adsl,
  id_cols = c("USUBJID", "SUBJID"),
  treatment_cols = c("TRT01A", "TRT01AN")
))
occds_summary <- glimpse_occds(
  adae,
  syn_adsl,
  id_cols = "USUBJID",
  seq_col = "AESEQ"
)
syn_adae <- simulate_occds(occds_summary, seed = 42)
head(syn_adae)
```

---

simulate_study *Simulate synthetic ADaM datasets for a study.*

Description

Taking as input a yaml configuration that specifies the ADaM datasets for a study of interest, this function will simulate synthetic versions of each and save them as individual .rds files in a specified directory.

Usage

```
simulate_study(config_path)
```

Arguments

config_path	character - path to YAML configuration file.

Details

The YAML configuration file should have the following structure:

```
output_dir: "/path/to/output_directory"
seed: 32

datasets:
  adsl:
    dataset_type: "adsl"
    path: "/path/to/adsl.sas7bdat"
    id_cols: ["USUBJID", "SUBJID"]
    treatment_cols: ["TRT01A", "TRT01AN"]
    flag_cols: ["SAFFL", "ITTFL"]
```

```
  ordered_col_sets:
    - ["REGION1", "REGION1N"]

adlb:
  dataset_type: "bds"
  path: "/path/to/adlb.sas7bdat"
  id_cols: ["USUBJID"]
  param_cols: ["PARAM", "PARAMCD"]
  visit_cols: ["AVISIT", "AVISITN"]
  flag_cols: ["ANL01FL"]

adtte:
  dataset_type: "tte"
  path: "/path/to/adtte.sas7bdat"
  param_cols: ["PARAM", "PARAMCD"]
  censor_cols: ["CNSR", "EVNTDESC", "CNSDTDSC"]
  flag_cols: ["ANL01FL"]
  ordered_col_sets:
    - ["SRCDOM", "SRCVAR"]

adae:
  dataset_type: "occds"
  path: "/path/to/adae.sas7bdat"
  id_cols: ["USUBJID"]
  seq_col: "AESEQ"
  flag_cols: ["AOCCFL", "TRTEMFL"]
  ordered_col_sets:
    - ["AEBODSYS", "AEDECOD"]
```

Dataset names in the output files are determined by the YAML keys under `datasets:` (e.g., `syn_adsl.rds`, `syn_adlb.rds`). Exactly one dataset must have `dataset_type: "adsl"`. The `output_dir` directory will be created automatically if it doesn't exist.

Value

`NULL` (invisibly). Synthetic datasets are saved as individual `.rds` files named `syn_{key}.rds` in the `output_dir` directory, where `{key}` is the dataset name from the YAML config. Each dataset has a `synadam_version` attribute containing the package version.

Examples

```
out_dir <- file.path(tempdir(), "syn_study")
yaml_path <- generate_study_config(adam_dir, output_dir = out_dir)

# Simulate the whole study; synthetic syn_*.rds files land in out_dir.
simulate_study(yaml_path)
list.files(out_dir, pattern = "\\.rds$")
```

simulate_study_from_summary *Simulate synthetic ADaM datasets from a previously-saved study summary.*

Description

Reads a study summary written by `glimpse_study()` and produces synthetic datasets, decoupling simulation from the original `.sas7bdat` files.

Usage

```
simulate_study_from_summary(study_summary_path, output_dir, seed = NULL)
```

Arguments

study_summary_path	`character` - path to the `.rds` study summary written by `glimpse_study()`.
output_dir	`character` - directory to write `syn_{key}.rds` files to. Created if missing.
seed	`integer` or `NULL` - simulation seed. When `NULL` (default), the seed stored in the study summary is used; if the study summary has no seed, it falls back to `123`. Override to draw a different synthetic replicate from the same summaries.

Value

`NULL` (invisibly). Synthetic datasets are saved as `syn_{key}.rds` files in `output_dir`. Each dataset has a `synadam_version` attribute.

See Also

`glimpse_study()`, `simulate_study()`.

Examples

```
# Generate a config and glimpse the datasets into a study summary.
yaml_path <- generate_study_config(
  adam_dir,
  output_dir = file.path(tempdir(), "syn_data")
)
study_summary_path <- tempfile(fileext = ".rds")
glimpse_study(yaml_path, study_summary_path)

# Simulate synthetic datasets from the saved study summary.
out_dir <- file.path(tempdir(), "syn_data")
simulate_study_from_summary(study_summary_path, out_dir)
list.files(out_dir, pattern = "\\.rds$")
```

`simulate_tte` *Simulate a synthetic TTE dataset.*

Description

Generates synthetic TTE data by creating a spine from parameter combinations and ADSL subject records, then populating remaining columns, preserving the original data structure. Produces exactly one record per subject per parameter (the defining TTE constraint).

Usage

```
simulate_tte(tte_summary, seed = NULL)
```

Arguments

`tte_summary`	`summary_tte` - summary object created by `glimpse_tte()`.
`seed`	`integer` - random seed for reproducibility.

Value

`data.frame` - synthetic TTE dataset.

Examples

```
syn_adsl <- data.frame(
  USUBJID = as.character(1:5),
  TRT01A = c("Placebo", "Placebo", "Drug A", "Drug A", "Drug A"),
  TRT01AN = c(1, 1, 2, 2, 2)
)
tte <- data.frame(
  USUBJID = as.character(rep(1:3, each = 2)),
  PARAM = rep(c("Overall Survival", "Progression Free Survival"), 3),
  PARAMCD = rep(c("OS", "PFS"), 3),
  CNSR = c(0, 1, 0, 1, 1, 0),
  AVAL = c(365, 500, 180, 250, 600, 120)
)

tte_summary <- glimpse_tte(
  tte,
  syn_adsl,
  param_cols = c("PARAM", "PARAMCD")
)
syn_tte <- simulate_tte(tte_summary, seed = 42)
head(syn_tte)
```

Index