Package 'simaerep'

May 9, 2026

Title Detect Clinical Trial Sites Over- or Under-Reporting Clinical Events

Version 1.0.0

Description Monitoring reporting rates of subject-level clinical events (e.g. adverse events, protocol deviations) reported by clinical trial sites is an important aspect of risk-based quality monitoring strategy. Sites that are under-reporting or over-reporting events can be detected using bootstrap simulations during which patients are redistributed between sites. Site-specific distributions of event reporting rates are generated that are used to assign probabilities to the observed reporting rates. (Koneswarakantha 2024 <doi:10.1007/s43441-024-00631-8>).

URL https://openpharma.github.io/simaerep/, https://github.com/openpharma/simaerep/

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0), ggplot2

Imports dplyr (>= 1.1.0), tidyr (>= 1.1.0), magrittr, purrr, rlang, stringr, forcats, cowplot, RColorBrewer, furrr (>= 0.2.1), progressr, knitr, tibble, dbplyr, glue

Suggests testthat, devtools, pkgdown, spelling, haven, vdiffr, lintr, DBI, duckdb, ggExtra

RoxygenNote 7.3.2

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Bjoern Koneswarakantha [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-4585-7799>), F. Hoffmann-La Roche Ltd [cph]

Maintainer Bjoern Koneswarakantha <bjoern.koneswarakantha@roche.com>

Repository CRAN

Date/Publication 2025-10-28 11:40:02 UTC

Contents

`get_cum_mean_event_dev` *Get cumulative mean event development*

Description

Calculate average increase of events per visit and cumulative average increase.

Usage

```
get_cum_mean_event_dev(
  df_visit,
  group = c("site_number", "study_id"),
  event_names = c("ae")
)
```

Arguments

`df_visit`	Data frame with columns: study_id, site_number, patnum, visit, n_ae.
`group`	character, grouping variable, one of: c("site_number", "study_id")
`event_names`	vector, contains the event names, default = "event"

Details

This is more stable than using mean cumulative patient count per visit as only a few patients will contribute to later visits. Here the impact of the later visits is reduced as they can only add or subtract to the results from earlier visits and not shift the mean independently.

Examples

```
df_visit <- sim_test_data_study(n_pat = 1000, n_sites = 10) %>%
  dplyr::rename(
    site_number = site_id,
    patnum = patient_id,
    n_ae = n_event
  )

get_cum_mean_event_dev(df_visit)
get_cum_mean_event_dev(df_visit, group = "study_id")
```

---

get_portf_config *Get Portfolio Configuration*

Description

Get Portfolio configuration from a df_visit input dataframe. Will . filter studies with only a few sites and patients and will anonymize IDs. . Portfolio configuration can be used by sim_test_data_portfolio to generate data for an artificial portfolio.

Usage

```
get_portf_config(
  df_visit,
  check = TRUE,
  min_pat_per_study = 100,
  min_sites_per_study = 10,
  anonymize = TRUE,
  pad_width = 4
)
```

Arguments

df_visit	input dataframe with columns study_id, site_id, patient_id, visit, n_events. Can also be a lazy database table.
check	logical, perform standard checks on df_visit, Default: TRUE
min_pat_per_study	minimum number of patients per study, Default: 100
min_sites_per_study	minimum number of sites per study, Default: 10
anonymize	logical, Default: TRUE
pad_width	padding width for newly created IDs, Default: 4

Value

dataframe with the following columns:

study_id study identification

event_per_visit_mean mean event per visit per study

site_id site

max_visit_sd standard deviation of maximum patient visits per site

max_visit_mean mean of maximum patient visits per site

n_pat number of patients

See Also

sim_test_data_study get_portf_config sim_test_data_portfolio

Examples

```
df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.4, factor_event_rate = - 0.6,
                                 study_id = "A")


df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.2, factor_event_rate = - 0.1,
                                 study_id = "B")


df_visit <- dplyr::bind_rows(df_visit1, df_visit2)


get_portf_config(df_visit)


# Database example
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = ":memory:")
dplyr::copy_to(con, df_visit, "visit")
tbl_visit <- dplyr::tbl(con, "visit")
get_portf_config(tbl_visit)
DBI::dbDisconnect(con)
```

---

get_portf_event_rates *Get Portfolio Event Rates Calculates mean event rates per study and visit in a df_visit simaerep input dataframe.*

Description

Get Portfolio Event Rates Calculates mean event rates per study and visit in a df_visit simaerep input dataframe.

Usage

```
get_portf_event_rates(df_visit, check = TRUE, anonymize = TRUE, pad_width = 4)
```

Arguments

df_visit	input dataframe with columns study_id, site_id, patient_id, visit, n_events. Can also be a lazy database table.
check	logical, perform standard checks on df_visit, Default: TRUE
anonymize	logical, Default: TRUE
pad_width	padding width for newly created IDs, Default: 4

Examples

```
df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.4, factor_event_rate = - 0.6,
                                 study_id = "A")


df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.2, factor_event_rate = - 0.1,
                                 study_id = "B")


df_visit <- dplyr::bind_rows(df_visit1, df_visit2)


get_portf_event_rates(df_visit)


# Database example
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = ":memory:")
dplyr::copy_to(con, df_visit, "visit")
tbl_visit <- dplyr::tbl(con, "visit")
get_portf_event_rates(tbl_visit)
DBI::dbDisconnect(con)
```

---

orivisit *create orivisit object*

Description

Internal S3 object, stores lazy reference to original visit data.

Usage

```
orivisit(
  df_visit,
  call = NULL,
  env = parent.frame(),
  event_names = c("event"),
 col_names = list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id",
    visit = "visit")
)
```

Arguments

df_visit	Data frame with columns: study_id, site_number, patnum, visit, n_ae.
call	optional, provide call, Default: NULL
env	Optional, provide environment of original visit data. Default: parent.frame().
event_names	vector, contains the event names, default = "event"
col_names	named list, indicate study_id, site_id, patient_id and visit column in df_visit input dataframe. Default: list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id", visit = "visit")

Details

Saves variable name of original visit data, checks whether it can be retrieved from parent environment and stores summary. Original data can be retrieved using as.data.frame(x).

Value

orivisit object

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = - 0.6
)#'

visit <- orivisit(df_visit)

object.size(df_visit)
object.size(visit)

as.data.frame(visit)
```

`plot.simaerep` *plot AE under-reporting simulation results*

Description

generic plot function for simaerep objects

Usage

```
## S3 method for class 'simaerep'
plot(
  x,
  ...,
  study = NULL,
  what = c("prob", "med75"),
  n_sites = 16,
  df_visit = NULL,
  env = parent.frame(),
  plot_event = x$event_names[1]
)
```

Arguments

`x`	simaerep object
`...`	additional parameters passed to plot_study() or plot_visit_med75()
`study`	character specifying study to be plotted, Default: NULL
`what`	one of c("ur", "med75"), specifying whether to plot site AE under-reporting or visit_med75 values, Default: 'ur'
`n_sites`	number of sites to plot, Default: 16
`df_visit`	optional, pass original visit data if it cannot be retrieved from parent environment, Default: NULL
`env`	optional, pass environment from which to retrieve original visit data, Default: parent.frame()
`plot_event`	vector containing the events that should be plotted, default = "ae"

Details

see plot_study() and plot_visit_med75()

Value

ggplot object

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = - 0.6
)

evrep <- simaerep(df_visit)

plot(evrep, what = "prob", study = "A")
plot(evrep, what = "med75", study = "A")
```

---

plot_dots *Plots AE per site as dots.*

Description

This plot is meant to supplement the package documentation.

Usage

```
plot_dots(
  df,
  nrow = 10,
  ncols = 10,
  col_group = "site",
  thresh = NULL,
  color_site_a = "#BDBDBD",
  color_site_b = "#757575",
  color_site_c = "gold3",
  color_high = "#00695C",
  color_low = "#25A69A",
  size_dots = 10
)
```

Arguments

df	dataframe, cols = c('site', 'patients', 'n_ae')
nrow	integer, number of rows, Default: 10
ncols	integer, number of columns, Default: 10
col_group	character, grouping column, Default: 'site'
thresh	numeric, threshold to determine color of mean_ae annotation, Default: NULL
color_site_a	character, hex color value, Default: '#BDBDBD'
color_site_b	character, hex color value, Default: '#757575'

color_site_c	character, hex color value, Default: 'gold3'
color_high	character, hex color value, Default: '#00695C'
color_low	character, hex color value, Default: '#25A69A'
size_dots	integer, Default: 10

Value

ggplot object

Examples

```
study <- tibble::tibble(
  site = LETTERS[1:3],
  patients = c(list(seq(1, 50, 1)), list(seq(1, 40, 1)), list(seq(1, 10, 1)))
) %>%
  tidyr::unnest(patients) %>%
  dplyr::mutate(n_ae = as.integer(runif(min = 0, max = 10, n = nrow(.))))

plot_dots(study)
```

---

plot_sim_example *Plot simulation example.*

Description

This plots supplements the package documentation.

Usage

```
plot_sim_example(
  substract_ae_per_pat = 0,
  size_dots = 10,
  size_raster_label = 12,
  color_site_a = "#BDBDBD",
  color_site_b = "#757575",
  color_site_c = "gold3",
  color_high = "#00695C",
  color_low = "#25A69A",
  title = TRUE,
  legend = TRUE,
  seed = 5
)
```

Arguments

substract_ae_per_pat	integer, subtract aes from patients at site C, Default: 0
size_dots	integer, Default: 10
size_raster_label	integer, Default: 12
color_site_a	character, hex color value, Default: '#BDBDBD'
color_site_b	character, hex color value, Default: '#757575'
color_site_c	character, hex color value, Default: 'gold3'
color_high	character, hex color value, Default: '#00695C'
color_low	character, hex color value, Default: '#25A69A'
title	logical, include title, Default: T
legend	logical, include legend, Default: T
seed	pass seed for simulations Default: 5

Details

uses `plot_dots()` and adds 2 simulation panels, uses made-up site config with three sites A,B,C simulating site C

Value

ggplot

See Also

`get_legend`,`plot_grid`

Examples

```
plot_sim_example(size_dots = 5)
```

---

purrr_bar *Execute a purrr or furrr function with a progress bar.*

Description

Internal utility function.

Usage

```
purrr_bar(
  ...,
  .purrr,
  .f,
  .f_args = list(),
  .purrr_args = list(),
  .steps,
  .slow = FALSE,
  .progress = TRUE
)
```

Arguments

...	iterable arguments passed to .purrr
.purrr	purrr or furrr function
.f	function to be executed over iterables
.f_args	list of arguments passed to .f, Default: list()
.purrr_args	list of arguments passed to .purrr, Default: list()
.steps	integer number of iterations
.slow	logical slows down execution, Default: FALSE
.progress	logical, show progress bar, Default: TRUE

Details

Call still needs to be wrapped in `with_progress` or `with_progress_cnd()`

Value

result of function passed to .f

Examples

```
# purrr::map
progressr::with_progress(
  purrr_bar(rep(0.25, 5), .purrr = purrr::map, .f = Sys.sleep, .steps = 5)
)


# purrr::walk
progressr::with_progress(
 purrr_bar(rep(0.25, 5), .purrr = purrr::walk,.f = Sys.sleep, .steps = 5)
)

# progress bar off
progressr::with_progress(
  purrr_bar(
    rep(0.25, 5), .purrr = purrr::walk,.f = Sys.sleep, .steps = 5, .progress = FALSE
```

```
  )
)

# purrr::map2
progressr::with_progress(
  purrr_bar(
    rep(1, 5), rep(2, 5),
    .purrr = purrr::map2,
    .f = `+`,
    .steps = 5,
    .slow = TRUE
 )
)

# purrr::pmap
progressr::with_progress(
  purrr_bar(
    list(rep(1, 5), rep(2, 5)),
    .purrr = purrr::pmap,
    .f = `+`,
    .steps = 5,
    .slow = TRUE
 )
)

# define function within purr_bar() call
progressr::with_progress(
  purrr_bar(
    list(rep(1, 5), rep(2, 5)),
    .purrr = purrr::pmap,
    .f = function(x, y) {
      paste0(x, y)
    },
    .steps = 5,
    .slow = TRUE
 )
)

# with mutate
progressr::with_progress(
 tibble::tibble(x = rep(0.25, 5)) %>%
  dplyr::mutate(x = purrr_bar(x, .purrr = purrr::map, .f = Sys.sleep, .steps = 5))
)
```

`simaerep` *Create simaerep object*

Description

Simulate AE under-reporting probabilities.

Usage

```
simaerep(
  df_visit,
  r = 1000,
  check = TRUE,
  under_only = FALSE,
  visit_med75 = FALSE,
  inframe = TRUE,
  progress = TRUE,
  mult_corr = TRUE,
  poisson_test = FALSE,
  env = parent.frame(),
  event_names = c("event"),
 col_names = list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id",
    visit = "visit")
)

simaerep_inframe(
  df_visit,
  r = 1000,
  under_only = FALSE,
  visit_med75 = FALSE,
  check = TRUE,
  env = parent.frame(),
  event_names = c("event"),
  mult_corr = FALSE,
 col_names = list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id",
    visit = "visit")
)

simaerep_classic(
  df_visit,
  check = TRUE,
  progress = TRUE,
  env = parent.frame(),
  under_only = TRUE,
  r = 1000,
  mult_corr = FALSE,
  poisson_test = FALSE,
  event_names = "event",
 col_names = list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id",
    visit = "visit")
)
```

Arguments

df_visit	Data frame with columns: study_id, site_number, patnum, visit, n_ae.
r	Integer or tbl_object, number of repetitions for bootstrap simulation. Pass a

	tbl object referring to a table with one column and as many rows as desired repetitions. Default: 1000.
`check`	Logical, perform data check and attempt repair with `check_df_visit()`. Computationally expensive on large data sets. Default: TRUE.
`under_only`	Logical, compute under-reporting probabilities only. only applies to the classic algorithm in which a one-sided evaluation can save computation time. Default: FALSE
`visit_med75`	Logical, should evaluation point visit_med75 be used. Compatible with inframe and classic version of the algorithm. Default: FALSE
`inframe`	Logical, when FALSE classic simaerep algorithm will be used. The default inframe method uses only table operations and is compatible with dbplyr supported database backends. Default: TRUE
`progress`	Logical, display progress bar. Default: TRUE.
`mult_corr`	Logical, multiplicity correction, Default: TRUE
`poisson_test`	logical, compute p-value with poisson test, only supported by the classic algorithm using visit_med75. Default: FALSE
`env`	Optional, provide environment of original visit data. Default: parent.frame().
`event_names`	vector, contains the event names, default = "event"
`col_names`	named list, indicate study_id, site_id, patient_id and visit column in df_visit input dataframe. Default: list(study_id = "study_id", site_id = "site_id", patient_id = "patient_id", visit = "visit")

Details

Executes `site_aggr()`, `sim_sites()`, and `eval_sites()` on original visit data and stores all intermediate results. Stores lazy reference to original visit data for facilitated plotting using generic plot(x).

Value

A simaerep object. Results are contained in the attached df_eval dataframe.

Column Name	Description	Type
study_id	The study ID	Character
site_id.	The site ID	Character
(event)_count	Site event count	Numeric
(event)_per_visit_site	Site Ratio of event count divided by visits	Numeric
visits	Site visit count	Numeric
n_pat	Site patient count	Numeric
(event)_per_visit_study	Simulated study ratio	Numeric
(event)_prob	Site event ratio probability from -1 to 1	Numeric
(event)_delta	Difference expected vs reported events	Numeric

See Also

site_aggr, sim_sites, eval_sites, orivisit, plot.simaerep, print.simaerep, simaerep_inframe

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = - 0.6
)

evrep <- simaerep(df_visit)
evrep
str(evrep)

# simaerep classic algorithm

evrep <- simaerep(df_visit, inframe = FALSE, under_only = TRUE, mult_corr = TRUE)
evrep

# multiple events

df_visit_events_test <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = - 0.6,
  event_rates = list(0.5, 0.3),
  event_names = c("ae", "pd")
)

evsrep <- simaerep(df_visit_events_test, inframe = TRUE, event_names = c("ae", "pd"))

evsrep


# Database example
con <- DBI::dbConnect(duckdb::duckdb(), dbdir = ":memory:")
df_r <- tibble::tibble(rep = seq(1, 1000))
dplyr::copy_to(con, df_visit, "visit")
dplyr::copy_to(con, df_r, "r")
tbl_visit <- dplyr::tbl(con, "visit")
tbl_r <- dplyr::tbl(con, "r")
simaerep(tbl_visit, r = tbl_r)
DBI::dbDisconnect(con)
```

`sim_inframe` *Calculate prob for study sites using table operations*

Description

Calculate prob for study sites using table operations

Usage

```
sim_inframe(df_visit, r = 1000, df_site = NULL, event_names = c("ae"))
```

Arguments

df_visit	Data frame with columns: study_id, site_number, patnum, visit, n_ae.
r	Integer or tbl_object, number of repetitions for bootstrap simulation. Pass a tbl object referring to a table with one column and as many rows as desired repetitions. Default: 1000.
df_site	dataframe as returned be site_aggr(), Will switch to visit_med75. Default: NULL
event_names	vector, contains the event names, default = "event"

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = - 0.6
) %>%
dplyr::rename(
  site_number = site_id,
  patnum = patient_id,
  n_ae = n_event
)

df_sim <- simaerep:::sim_inframe(df_visit)
```

---

sim_out *simulate under-reporting*

Description

we remove a fraction of events from a specific site

Usage

```
sim_out(df_visit, study_id, site_id, factor_event)
```

Arguments

df_visit	dataframe
study_id	character
site_id	character
factor_event	double, negative values for under-reporting positive for for over-reporting.

Details

we determine the absolute number of events per patient for removal. Then them remove them at the first visit. We intentionally allow fractions

Examples

```
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 10)


df_ur <- sim_out(df_visit, "A", site_id = "S0001", factor_event = - 0.35)

# Example cumulated event for first patient with 35% under-reporting
df_ur[df_ur$site_id == "S0001" & df_ur$patient_id == "P000001",]$n_event

# Example cumulated event for first patient with no under-reporting
df_visit[df_visit$site_id == "S0001" & df_visit$patient_id == "P000001",]$n_event
```

---

sim_sites *Calculate prob_lower and poisson.test pvalue for study sites.*

Description

Collects the number of AEs of all eligible patients that meet visit_med75 criteria of site. Then calculates poisson.test pvalue and bootstrapped probability of having a lower mean value. Used by simaerep_classic()

Usage

```
sim_sites(
  df_site,
  df_visit,
  r = 1000,
  poisson_test = TRUE,
  prob_lower = TRUE,
  progress = TRUE,
  under_only = TRUE
)
```

Arguments

df_site	dataframe created by site_aggr
df_visit	dataframe, created by sim_sites
r	integer, denotes number of simulations, default = 1000
poisson_test	logical, calculates poisson.test pvalue
prob_lower	logical, calculates probability for getting a lower value

progress	logical, display progress bar, Default = TRUE
under_only	compute under-reporting probabilities only, default = TRUE check_df_visit(), computationally expensive on large data sets. Default: TRUE

Value

dataframe with the following columns:

study_id study identification

site_number site identification

n_pat number of patients at site

visit_med75 median(max(visit)) * 0.75

n_pat_with_med75 number of patients at site with med75

mean_ae_site_med75 mean AE at visit_med75 site level

mean_ae_study_med75 mean AE at visit_med75 study level

n_pat_with_med75_study number of patients at study with med75 excl. site

pval p-value as returned by poisson.test

prob_low bootstrapped probability for having mean_ae_site_med75 or lower

See Also

sim_sites, site_aggr, pat_pool, prob_lower_site_ae_vs_study_ae, poiss_test_site_ae_vs_study_ae, sim_sites, prep_for_sim simaerep_classic

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = 0.6
  ) %>%
  # internal functions require internal column names
  dplyr::rename(
    n_ae = n_event,
    site_number = site_id,
    patnum = patient_id
  )

df_site <- site_aggr(df_visit)

df_sim_sites <- sim_sites(df_site, df_visit, r = 100)

df_sim_sites %>%
 knitr::kable(digits = 2)
```

`sim_test_data_portfolio` *Simulate Portfolio Test Data*

Description

Simulate visit level data from a portfolio configuration.

Usage

```
sim_test_data_portfolio(
  df_config,
  df_event_rates = NULL,
  progress = TRUE,
  parallel = TRUE
)
```

Arguments

`df_config`	dataframe as returned by `get_portf_config`
`df_event_rates`	dataframe with event rates. Default: NULL
`progress`	logical, Default: TRUE
`parallel`	logical activate parallel processing, see details, Default: FALSE

Details

uses `sim_test_data_study`. We use the `furrr` package to implement parallel processing as these simulations can take a long time to run. For this to work we need to specify the plan for how the code should run, e.g. 'plan(multisession, workers = 3)

Value

dataframe with the following columns:

study_id study identification

event_per_visit_mean mean event per visit per study

site_id site

max_visit_sd standard deviation of maximum patient visits per site

max_visit_mean mean of maximum patient visits per site

patient_id number of patients

visit visit number

n_event cumulative sum of events

See Also

`sim_test_data_study` `get_portf_config` `sim_test_data_portfolio`

Examples

```
df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.4, factor_event_rate = 0.6,
                                 study_id = "A")

df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 ratio_out = 0.2, factor_event_rate = 0.1,
                                 study_id = "B")


df_visit <- dplyr::bind_rows(df_visit1, df_visit2)

df_config <- get_portf_config(df_visit)

df_config

df_portf <- sim_test_data_portfolio(df_config)

df_portf
```

`sim_test_data_study` *simulate study test data*

Description

evenly distributes a number of given patients across a number of given sites. Then simulates event reporting of each patient reducing the number of reported events for patients distributed to event-under-reporting sites.

Usage

```
sim_test_data_study(
  n_pat = 1000,
  n_sites = 20,
  ratio_out = 0,
  factor_event_rate = 0,
  max_visit_mean = 20,
  max_visit_sd = 4,
  event_rates = dgamma(seq(1, 20, 0.5), shape = 5, rate = 2) * 5 + 0.1,
  event_names = c("event"),
  study_id = "A"
)
```

Arguments

`n_pat`	integer, number of patients, Default: 1000

n_sites	integer, number of sites, Default: 20
ratio_out	ratio of sites with outlier, Default: 0
factor_event_rate	event reporting rate factor for site outlier, will modify mean event per visit rate used for outlier sites. Negative Values will simulate under-reporting, positive values over-reporting, e.g. -0.4 -> 40% under-reporting, +0.4 -> 40% over-reporting Default: 0
max_visit_mean	mean of the maximum number of visits of each patient, Default: 20
max_visit_sd	standard deviation of maximum number of visits of each patient, Default: 4
event_rates	list or vector with visit-specific event rates. Use list for multiple event names, Default: dgamma(seq(1, 20, 0.5), shape = 5, rate = 2) * 5 + 0.1
event_names	vector, contains the event names, default = "event"
study_id	character, Default: "A"

Details

maximum visit number will be sampled from normal distribution with characteristics derived from max_visit_mean and max_visit_sd, while the events per visit will be sampled from a poisson distribution described by events_per_visit_mean.

Value

tibble with columns site_id, patient_id, is_out, max_visit_mean, max_visit_sd, event_per_visit_mean, visit, n_event

Examples

```
set.seed(1)
# no outlier
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5)
df_visit[which(df_visit$patient_id == "P000001"),]

# under-reporting outlier
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5,
    ratio_out = 0.2, factor_event_rate = -0.5)
df_visit[which(df_visit$patient_id == "P000001"),]

# constant event rates
sim_test_data_study(n_pat = 100, n_sites = 5, event_rates = 0.5)

# non-constant event rates for two event types
event_rates_ae <- c(0.7, rep(0.5, 8), rep(0.3, 5))
event_rates_pd <- c(0.3, rep(0.4, 6), rep(0.1, 5))

sim_test_data_study(
n_pat = 100,
n_sites = 5,
event_names = c("ae", "pd"),
event_rates = list(event_rates_ae, event_rates_pd)
```

```
)
```

`site_aggr` *Aggregate from visit to site level.*

Description

Calculates visit_med75, n_pat_with_med75 and mean_ae_site_med75. Used by `simaerep_classic()`

Usage

```
site_aggr(
  df_visit,
  method = "med75_adj",
  min_pat_pool = 0.2,
  event_names = c("ae")
)
```

Arguments

`df_visit`	dataframe with columns: study_id, site_number, patnum, visit, n_ae
`method`	character, one of c("med75", "med75_adj", "max") defining method for defining evaluation point visit_med75 (see details), Default: "med75_adj"
`min_pat_pool`	double, minimum ratio of available patients available for sampling. Determines maximum visit_med75 value see Details. Default: 0.2
`event_names`	vector, contains the event names, default = "ae"

Details

For determining the visit number at which we are going to evaluate AE reporting we take the maximum visit of each patient at the site and take the median. Then we multiply with 0.75 which will give us a cut-off point determining which patient will be evaluated. Of those patients we will evaluate we take the minimum of all maximum visits hence ensuring that we take the highest visit number possible without excluding more patients from the analysis. In order to ensure that the sampling pool for that visit is large enough we limit the visit number by the 80% quantile of maximum visits of all patients in the study. "max" will determine site max visit, flag patients that concluded max visit and count patients and patients that concluded max visit.

Value

dataframe with the following columns:

study_id study identification

site_number site identification

n_pat number of patients, site level

visit_med75 adjusted median(max(visit)) * 0.75 see Details

n_pat_with_med75 number of patients that meet visit_med75 criterion, site level

mean_ae_site_med75 mean AE at visit_med75, site level

See Also

simaerep_classic()

Examples

```
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  ratio_out = 0.4,
  factor_event_rate = 0.6
  ) %>%
  # internal functions require internal column names
  dplyr::rename(
    n_ae = n_event,
    site_number = site_id,
    patnum = patient_id
  )

df_site <- site_aggr(df_visit)

df_site %>%
  knitr::kable(digits = 2)
```

---

with_progress_cnd *Conditional* with_progress.

Description

Internal function. Use instead of with_progress within custom functions with progress bars.

Usage

```
with_progress_cnd(ex, progress = TRUE)
```

Arguments

ex	expression
progress	logical, Default: TRUE

Details

This wrapper adds a progress parameter to with_progress so that we can control the progress bar in the user facing functions. The progressbar only shows in interactive mode.

Value

No return value, called for side effects

See Also

with_progress

Examples

```
if (interactive()) {

 with_progress_cnd(
   purrr_bar(rep(0.25, 5), .purrr = purrr::map, .f = Sys.sleep, .steps = 5),
   progress = TRUE
 )

 with_progress_cnd(
   purrr_bar(rep(0.25, 5), .purrr = purrr::map, .f = Sys.sleep, .steps = 5),
   progress = FALSE
 )

# wrap a function with progress bar with another call with progress bar

f1 <- function(x, progress = TRUE) {
  with_progress_cnd(
   purrr_bar(x, .purrr = purrr::walk, .f = Sys.sleep, .steps = length(x), .progress = progress),
    progress = progress
  )
}

# inner progress bar blocks outer progress bar
progressr::with_progress(
  purrr_bar(
    rep(rep(1, 3),3), .purrr = purrr::walk, .f = f1, .steps = 3,
    .f_args = list(progress = TRUE)
  )
)

# inner progress bar turned off
progressr::with_progress(
  purrr_bar(
    rep(list(rep(0.25, 3)), 5), .purrr = purrr::walk, .f = f1, .steps = 5,
    .f_args = list(progress = FALSE)
  )
)
}
```

Index