Package ‘simaerep’

March 3, 2023

Title  Find Clinical Trial Sites Under-Reporting Adverse Events

Version  0.4.3

Description  Monitoring of Adverse Event (AE) reporting in clinical trials is important for patient safety. Sites that are under-reporting AEs can be detected using Bootstrap-based simulations that simulate overall AE reporting. Based on the simulation an AE under-reporting probability is assigned to each site in a given trial (Koneswarakantha 2021 <doi:10.1007/s40264-020-01011-5>).


License  MIT + file LICENSE

Encoding  UTF-8

Depends  R (>= 4.0), ggplot2

Imports  dplyr (>= 1.0.0), tidyr (>= 1.1.0), magrittr, purrr, rlang, stringr, forcats, cowplot, RColorBrewer, furrr (>= 0.2.1), progressr, knitr, tibble

Suggests  testthat, devtools, pkgdown, spelling, haven, vdiffir, lintr

RoxygenNote  7.2.0

Language  en-US

Config/testthat/edition  3

NeedsCompilation  no

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

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

Repository  CRAN

Date/Publication  2023-03-03 10:30:02 UTC
R topics documented:

aggr_duplicated_visits .................................................. 3
check_df_visit .............................................................. 3
eval_sites ................................................................. 4
eval_sites_deprecated ...................................................... 5
exp_implicit_missing_visits ............................................. 7
get_config ................................................................. 7
get_ecd_values ............................................................ 9
get_pat_pool_config ...................................................... 10
get_portf_perf ............................................................ 11
get_site_mean_ae_dev ..................................................... 12
get_visit_med75 .......................................................... 13
is_orivisit ............................................................... 14
is_simaerep .............................................................. 14
orivisit ................................................................. 15
pat_aggr ................................................................. 16
pat_pool ................................................................. 16
plot.simaerep ........................................................... 17
plot_dots ................................................................. 18
plot_sim_example ........................................................ 19
plot_sim_examples ....................................................... 21
plot_study .............................................................. 21
plot_visit_med75 ........................................................ 23
poiss_test_site_ae_vs_study_ae ....................................... 24
prep_for_sim .............................................................. 25
prob_lower_site_ae_vs_study_ae ....................................... 26
purrr_bar ................................................................. 27
simaerep .............................................................. 29
sim_after_prep ........................................................... 30
sim_scenario ............................................................ 31
sim_sites ................................................................. 32
sim_studies .............................................................. 33
sim_test_data_patient .................................................. 35
sim_test_data_portfolio ............................................... 36
sim_test_data_study .................................................... 37
sim_ur_scenarios ......................................................... 39
site_aggr ............................................................... 41
with_progress_cnd ...................................................... 42

Index 45
aggr_duplicated_visits

Aggregate duplicated visits.

Description

Internal function called by `check_df_visit()`.

Usage

```r
aggr_duplicated_visits(df_visit)
```

Arguments

- `df_visit` : dataframe with columns: study_id, site_number, patnum, visit, n_ae

Value

- `df_visit` corrected

check_df_visit

Integrity check for `df_visit`.

Description

Internal function used by all functions that accept `df_visit` as a parameter. Checks for NA columns, numeric visits and AEs, implicitly missing and duplicated visits.

Usage

```r
check_df_visit(df_visit)
```

Arguments

- `df_visit` : dataframe with columns: study_id, site_number, patnum, visit, n_ae

Value

- corrected `df_visit`
Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.6
)

df_visit$study_id <- "A"

df_visit_filt <- df_visit %>%
  dplyr::filter(visit != 3)

df_visit_corr <- check_df_visit(df_visit_filt)
3 %in% df_visit_corr$visit
nrow(df_visit_corr) == nrow(df_visit)

df_visit_corr <- check_df_visit(dplyr::bind_rows(df_visit, df_visit))
 nrow(df_visit_corr) == nrow(df_visit)
```

---

eval_sites

Evaluate sites.

Description

Correct under-reporting probabilities using `p.adjust`.

Usage

eval_sites(df_sim_sites, method = "BH", ...)

Arguments

- `df_sim_sites` dataframe generated by `sim_sites`
- `method` character, passed to `stats::p.adjust()`, if NULL `eval_sites_deprecated()` is used instead. Default = "BH"
- ... use to pass `r_sim_sites` parameter to `eval_sites_deprecated()`

Value

dataframe with the following columns:

- `study_id` study identification
- `site_number` site identification
- `visit_med75` median(max(visit)) * 0.75
### eval_sites_deprecated

Mean AE at visit_med75 site level

Mean AE at visit_med75 study level

P-value as returned by `poisson.test`

Bootstrapped probability for having mean_ae_site_med75 or lower

Adjusted p-values

Adjusted bootstrapped probability for having mean_ae_site_med75 or lower

Probability under-reporting as 1 - pval_adj, `poisson.test` (use as benchmark)

Probability under-reporting as 1 - prob_low_adj, bootstrapped (use)

---

### Description

Correct under-reporting probabilities by the expected number of false positives (fp). This has been deprecated in favor of more conventional methods available via `p.adjust`.

### Usage

`eval_sites_deprecated(df_sim_sites, r_sim_sites)`

### Arguments

- **df_sim_sites**: dataframe generated by `sim_sites()`
- **r_sim_sites**: integer, number of repeats for bootstrap resampling for site simulation, needed for zero probability correction for fp calculation, Default: 1000

---

### Examples

```r
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5, frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit$study_id <- "A"

df_site <- site_aggr(df_visit)

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

df_eval <- eval_sites(df_sim_sites)

# use deprecated method -------

df_eval <- eval_sites(df_sim_sites, method = NULL, r_sim_sites = 100)

df_eval
```

---

```
## eval_sites_deprecated  Evaluate sites.

**Description**

Correct under-reporting probabilities by the expected number of false positives (fp). This has been deprecated in favor of more conventional methods available via `p.adjust`.

**Usage**

`eval_sites_deprecated(df_sim_sites, r_sim_sites)`

**Arguments**

- **df_sim_sites**: dataframe generated by `sim_sites()`
- **r_sim_sites**: integer, number of repeats for bootstrap resampling for site simulation, needed for zero probability correction for fp calculation, Default: 1000
Details

If by chance expected number of false positives (fp) is greater than the total number of positives (p) we set \( p_{\text{vs}\_fp\_ratio} = 1 \) and \( \text{prob\_ur} = 0 \).

Value

dataframe with the following columns:

- **study_id**: study identification
- **site_number**: site identification
- **visit\_med75**: median(max(visit)) * 0.75
- **mean\_ae\_site\_med75**: mean AE at visit\_med75 site level
- **mean\_ae\_study\_med75**: mean AE at visit\_med75 study level
- **pval**: p-value as returned by `poisson.test`
- **prob\_low**: bootstrapped probability for having mean\_ae\_site\_med75 or lower
- **n\_site**: number of study sites
- **pval\_n\_detected**: sites with the same p-value or lower
- **pval\_fp**: expected number of fp, \( pval \times n\_site \)
- **pval\_p\_vs\_fp\_ratio**: odds under-reporting as \( p/\text{fp} \), \`poisson.test` (use as benchmark)
- **pval\_prob\_ur**: probability under-reporting as \( 1 - \text{fp}/p \), \`poisson.test` (use as benchmark)
- **prob\_low\_n\_detected**: sites with same bootstrapped probability or lower
- **prob\_low\_fp**: expected number of fp, prob\_lower \( \times n\_site \)
- **prob\_low\_p\_vs\_fp\_ratio**: odds under-reporting as \( p/\text{fp} \), bootstrapped (use)
- **prob\_low\_prob\_ur**: probability under-reporting as \( 1 - \text{fp}/p \), bootstrapped (use)

See Also

- `site_aggr()`, `sim_sites()`

Examples

```r
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5,
                                 frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit$study_id <- "A"
df_site <- site_aggr(df_visit)

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

df_eval <- eval_sites_deprecated(df_sim_sites, r_sim_sites = 100)
df_eval
```
exp_implicit_missing_visits

Expose implicitly missing visits.

Description

Internal function called by check_df_visit().

Usage

exp_implicit_missing_visits(df_visit)

Arguments

df_visit  dataframe with columns: study_id, site_number, patnum, visit, n_ae

Value

df_visit corrected

get_config  Get Portfolio Configuration

Description

Get Portfolio configuration from a dataframe aggregated on patient level with max_ae and max_visit. 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_config(
  df_site,
  min_pat_per_study = 100,
  min_sites_per_study = 10,
  anonymize = TRUE,
  pad_width = 4
)
Arguments

- `df_site` dataframe aggregated on patient level with `max_ae` and `max_visit`
- `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
- `ae_per_visit_mean` mean AE per visit per study
- `site_number` 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_config`
- `sim_test_data_portfolio`
- `sim_ur_scenarios`
- `get_portf_perf`

Examples

```r
library(dplyr)

df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10, 
   frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit1$study_id <- "A"

df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10, 
   frac_site_with_ur = 0.2, ur_rate = 0.1)

df_visit2$study_id <- "B"

df_visit <- bind_rows(df_visit1, df_visit2)

df_site_max <- df_visit %>%
   group_by(study_id, site_number, patnum) %>%
   summarise(max_visit = max(visit),
             max_ae = max(n_ae),
             .groups = "drop")

df_config <- get_config(df_site_max)

df_config
```
df_portf <- sim_test_data_portfolio(df_config)

df_portf

df_scen <- sim_ur_scenarios(df_portf,
extra_ur_sites = 2,
ur_rate = c(0.5, 1))

df_scen

df_perf <- get_portf_perf(df_scen)

df_perf

get_ecd_values

Get empirical cumulative distribution values of pval or prob_lower

Description

Test function, test applicability of poisson test, by calculating

• the bootstrapped probability of obtaining a specific p-value or lower, use in combination with sim_studies().

Usage

get_ecd_values(df_sim_studies, df_sim_sites, val_str)

Arguments

df_sim_studies dataframe, generated by sim_studies()

df_sim_sites dataframe, generated by sim_sites()

val_str c("prob_low","pval")

Details

trains a ecdf function for each studies based on the results of sim_studies()

Value

dataframe with the following columns:

study_id study identification

site_number site identification

visit_med75 median(max(visit)) * 0.75
mean_ae_site_med75  mean AE at visit_med75 site level
mean_ae_study_med75  mean AE at visit_med75 study level
pval/prob_low    p-value as returned by poisson.test
pval/prob_low_ecd p-value as returned by poisson.test

Examples

df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5,
    frac_site_with_ur = 0.4, ur_rate = 0.3)

    df_visit$study_id <- "A"
    df_site <- site_aggr(df_visit)

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

    df_sim_studies <- sim_studies(
        df_site = df_site,
        df_visit = df_visit,
        r = 3,
        parallel = FALSE,
        poisson_test = TRUE,
        prob_lower = TRUE
    )

    get_ecd_values(df_sim_studies, df_sim_sites, "prob_low")
    get_ecd_values(df_sim_studies, df_sim_sites, "pval")

get_pat_pool_config  Configure study patient pool by site parameters.

Description

Internal Function used by sim_sites()

Usage

get_pat_pool_config(df_visit, df_site, min_n_pat_with_med75 = 1)

Arguments

    df_visit     dataframe
    df_site      dataframe as created by site_aggr()
    min_n_pat_with_med75
        minimum number of patients with visit_med_75 for simulation, Default: 1

Details

For simulating a study we need to configure the study patient pool to match the configuration of the sites
**get_portf_perf**

**Value**

dataframe

**Examples**

df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 5,  
frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit1$study_id <- "A"

df_visit2 <- sim_test_data_study(n_pat = 1000, n_sites = 3,  
frac_site_with_ur = 0.2, ur_rate = 0.1)

df_visit2$study_id <- "B"

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

df_site <- site_aggr(df_visit)

df_config <- get_pat_pool_config(df_visit, df_site)

df_config

---

**get_portf_perf**  
*Get Portfolio Performance*

**Description**

Performance as true positive rate (tpr as tp/P) on the basis of desired false positive rates (fpr as fp/P).

**Usage**

get_portf_perf(df_scen, stat = "prob_low_prob_ur", fpr = c(0.001, 0.01, 0.05))

**Arguments**

df_scen  
dataframe as returned by `sim_ur_scenarios`

stat  
character denoting the column name of the under-reporting statistic, Default: 'prob_low_prob_ur'

fpr  
numeric vector specifying false positive rates, Default: c(0.001, 0.01, 0.05)

**Details**

DETAILS

**Value**

dataframe
get_site_mean_ae_dev

See Also

sim_test_data_study get_config sim_test_data_portfolio sim_ur_scenarios get_portf_perf

Examples

def_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 frac_site_with_ur = 0.4, ur_rate = 0.6)
def_visit1$study_id <- "A"
def_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                                 frac_site_with_ur = 0.2, ur_rate = 0.1)
def_visit2$study_id <- "B"
def_visit <- dplyr::bind_rows(df_visit1, df_visit2)
def_site_max <- df_visit %>%
  dplyr::group_by(study_id, site_number, patnum) %>%
  dplyr::summarise(max_visit = max(visit),
                   max_ae = max(n_ae),
                   .groups = "drop")
def_config <- get_config(df_site_max)
def_config
def_portf <- sim_test_data_portfolio(df_config)
def_portf
def_scen <- sim_ur_scenarios(df_portf,
                            extra_ur_sites = 2,
                            ur_rate = c(0.5, 1))
def_scen
def_perf <- get_portf_perf(df_scen)
def_perf

get_site_mean_ae_dev  Get site mean ae development.

Description

Internal function used by site_aggr(), plot_visit_med75(), returns mean AE development from visit 0 to visit_med75.
get_visit_med75

**Usage**

```r
get_site_mean_ae_dev(df_visit, df_pat, df_site)
```

**Arguments**

- `df_visit` dataframe
- `df_pat` dataframe as returned by `pat_aggr()`
- `df_site` dataframe as returned by `site_aggr()`

**Value**

dataframe

---

**get_visit_med75**

Get visit_med75.

---

**Description**

Internal function used by `site_aggr()`.

**Usage**

```r
get_visit_med75(df_pat, method = "med75_adj", min_pat_pool = 0.2)
```

**Arguments**

- `df_pat` dataframe as returned by `pat_aggr()`
- `method` character, one of `c("med75", "med75_adj")` 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

**Value**

dataframe
is_orivisit  is orivisit class

Description
internal function

Usage
is_orivisit(x)

Arguments
x       object

Value
logical

is_simaerep  is simaerep class

Description
internal function

Usage
is_simaerep(x)

Arguments
x       object

Value
logical
orivisit

create orivisit object

Description

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

Usage

orivisit(df_visit, call = NULL, env = parent.frame())

Arguments

df_visit  dataframe with original visit data

call       optional, provide call, Default: NULL

env        optional, provide environment of original visit data, Default: parent.frame()

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,
    frac_site_with_ur = 0.4,
    ur_rate = 0.6
)
df_visit$study_id <- "A"

visit <- orivisit(df_visit)

object.size(df_visit)

object.size(visit)

as.data.frame(visit)
pat_aggr

Aggregate visit to patient level.

Description

Internal function used by `site_aggr()` and `plot_visit_med75()`, adds the maximum visit for each patient.

Usage

```r
pat_aggr(df_visit)
```

Arguments

- `df_visit` : dataframe

Value

dataframe

---

pat_pool

Create a study specific patient pool for sampling

Description

Internal function for `sim_sites`, filter all visits greater than max_visit_med75_study returns dataframe with one column for studies and one column with nested patient data.

Usage

```r
pat_pool(df_visit, df_site)
```

Arguments

- `df_visit` : dataframe, created by `sim_sites`
- `df_site` : dataframe created by `site_aggr`

Value

dataframe with nested pat_pool column
Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.6
)

df_visit$study_id <- "A"

df_site <- site_aggr(df_visit)

df_pat_pool <- pat_pool(df_visit, df_site)

df_pat_pool
```

**plot.simaerep**

`plot AE under-reporting simulation results`

### Description

generic plot function for simaerep objects

### Usage

```r
## S3 method for class 'simaerep'
plot(
  x,
  ...,
  study = NULL,
  what = "ur",
  n_sites = 16,
  df_visit = NULL,
  env = parent.frame()
)
```

### 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_dots

Details

see plot_study() and plot_visit_med75()

Value

ggplot object

Examples

df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.6
)

df_visit$study_id <- "A"

aerep <- simaerep(df_visit)

plot(aerep, what = "ur", study = "A")
plot(aerep, 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
)
plot_sim_example

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

```r
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)
```

Description

This plots supplements the package documentation.

Usage

plot_sim_example(
  subtract_ae_per_pat = 0,
  size_dots = 10,
  size_raster_label = 12,
  color_site_a = "#BDBDBD",
  color_site_b = "#757575",
  color_site_c = "gold3",
  size_raster_label = 12,
  color_site_a = "#BDBDBD",
  color_site_b = "#757575",
  color_site_c = "gold3",
  size_dots = 10
)
plot_sim_example

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)
plot_sim_examples

Plot multiple simulation examples.

Description
This plot is meant to supplement the package documentation.

Usage

plot_sim_examples(subtract_ae_per_pat = c(0, 1, 3), ...)

Arguments

subtract_ae_per_pat
  integer, Default: c(0, 1, 3)
...
  parameters passed to plot_sim_example()

Details
This function is a wrapper for plot_sim_example()

Value

ggplot

See Also

ggdraw, draw_label, plot_grid

Examples

plot_sim_examples(size_dot = 3, size_raster_label = 10)
plot_sim_examples()

plot_study

Plot ae development of study and sites highlighting at risk sites.

Description
Most suitable visual representation of the AE under-reporting statistics.
Usage

plot_study(
  df_visit,
  df_site,
  df_eval,
  study,
  df_al = NULL,
  n_sites = 16,
  pval = FALSE
)

Arguments

df_visit       dataframe, created by sim_sites()
df_site        dataframe created by site_aggr()
df_eval        dataframe created by eval_sites()
study          study
df_al          dataframe containing study_id, site_number, alert_level_site, alert_level_study
                (optional), Default: NA
n_sites        integer number of most at risk sites, Default: 16
pval           logical show p-value, Default: FALSE

Details

Left panel shows mean AE reporting per site (lightblue and darkblue lines) against mean AE reporting of the entire study (golden line). Single sites are plotted in descending order by AE under-reporting probability on the right panel in which grey lines denote cumulative AE count of single patients. Grey dots in the left panel plot indicate sites that were picked for single plotting. AE under-reporting probability of dark blue lines crossed threshold of 95%. Numbers in the upper left corner indicate the ratio of patients that have been used for the analysis against the total number of patients. Patients that have not been on the study long enough to reach the evaluation point (visit_med75) will be ignored.

Value

ggplot

Examples

df_visit <- sim_test_data_study(n_pat = 1000, n_sites = 10,
                                frac_site_with_ur = 0.2, ur_rate = 0.15, max_visit_sd = 8)

df_visit$study_id <- "A"
df_site <- site_aggr(df_visit)

df_sim_sites <- sim_sites(df_site, df_visit, r = 100)
```r
plot_visit_med75 <- eval_sites(df_sim_sites)
plot_study(df_visit, df_site, df_eval, study = "A")
```

---

### plot_visit_med75

Plot patient visits against visit_med75.

#### Description

Plots cumulative AEs against visits for patients at sites of given study and compares against visit_med75.

#### Usage

```r
plot_visit_med75(
  df_visit,
  df_site = NULL,
  study_id_str,
  n_sites = 6,
  min_pat_pool = 0.2,
  verbose = TRUE
)
```

#### Arguments

- `df_visit`: dataframe
- `df_site`: dataframe, as returned by `site_aggr()`
- `study_id_str`: character, specify study in study_id column
- `n_sites`: integer, Default: 6
- `min_pat_pool`: double, minimum ratio of available patients available for sampling. Determines maximum visit_med75 value see Details. Default: 0.2
- `verbose`: logical, Default: TRUE

#### Value

ggplot

#### Examples

```r
df_visit <- sim_test_data_study(n_pat = 120, n_sites = 6,
                                 frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit$study_id <- "A"
df_site <- site_aggr(df_visit)

plot_visit_med75(df_visit, df_site, study_id_str = "A", n_site = 6)
```
Description

Internal function used by \texttt{sim\_sites()}. 

Usage

\texttt{poiss\_test\_site\_ae\_vs\_study\_ae(site\_ae, study\_ae, visit\_med75)}

Arguments

- \texttt{site\_ae} vector with AE numbers
- \texttt{study\_ae} vector with AE numbers
- \texttt{visit\_med75} integer

Details

sets \texttt{pvalue=1} if mean AE site is greater than mean AE study or \texttt{ttest} gives error

Value

\texttt{pval}

See Also

\texttt{sim\_sites()}

Examples

\begin{verbatim}
poiss_test_site_ae_vs_study_ae(
    site_ae = \texttt{c(5, 3, 3, 2, 1, 6)},
    study_ae = \texttt{c(9, 8, 7, 9, 6, 7, 8)},
    visit_med75 = 10
)

poiss_test_site_ae_vs_study_ae(
    site_ae = \texttt{c(11, 9, 8, 6, 3)},
    study_ae = \texttt{c(9, 8, 7, 9, 6, 7, 8)},
    visit_med75 = 10
)
\end{verbatim}
prep_for_sim

prep_for_sim Prepare data for simulation.

Description
Internal function called by sim_sites. Collect AEs per patient at visit_med75 for site and study as a vector of integers.

Usage
prep_for_sim(df_site, df_visit)

Arguments
- df_site dataframe created by site_aggr
- df_visit dataframe, created by sim_sites

Value
dataframe

See Also
sim_sites, sim_after_prep

Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.2
)

df_visit$study_id <- "A"

df_site <- site_aggr(df_visit)

df_prep <- prep_for_sim(df_site, df_visit)

df_prep
```

prob_lower_site_ae_vs_study_ae

*Calculate bootstrapped probability for obtaining a lower site mean AE number.*

---

**Description**

Internal function used by `sim_sites()`

**Usage**

```r
prob_lower_site_ae_vs_study_ae(site_ae, study_ae, r = 1000, parallel = FALSE)
```

**Arguments**

- `site_ae`: vector with AE numbers
- `study_ae`: vector with AE numbers
- `r`: integer, denotes number of simulations, default = 1000
- `parallel`: logical, toggles parallel processing on and off, default = F

**Details**

sets pvalue=1 if mean AE site is greater than mean AE study

**Value**

- `pval`

**See Also**

`safely`

**Examples**

```r
prob_lower_site_ae_vs_study_ae(
  site_ae = c(5, 3, 3, 2, 1, 6),
  study_ae = c(9, 8, 7, 9, 6, 7, 8),
  parallel = FALSE
)
```
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::walk(rep(0.25, 5), .purrr = purrr::walk, .f = Sys.sleep, .steps = 5)
)

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

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

# purrr::pmap
progressr::with_progress(
  purrr::pmap(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::pmap(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::walk(x, .purrr = purrr::map, .f = Sys.sleep, .steps = 5))
)
simaerep

create simaerep object

Description
simulate AE under-reporting probabilities

Usage
simaerep(
  df_visit,
  param_site_aggr = list(method = "med75_adj", min_pat_pool = 0.2),
  param_sim_sites = list(r = 1000, poisson_test = FALSE, prob_lower = TRUE),
  param_eval_sites = list(method = "BH"),
  progress = TRUE,
  check = TRUE,
  env = parent.frame()
)

Arguments
df_visit data frame with columns: study_id, site_number, patnum, visit, n_ae
param_site_aggr list of parameters passed to site_aggr(), Default: list(method = "med75_adj", min_pat_pool = 0.2)
param_sim_sites list of parameters passed to sim_sites(), Default: list(r = 1000, poisson_test = FALSE, prob_lower = TRUE)
param_eval_sites list of parameters passed to eval_sites(), Default: list(method = "BH")
progress logical, display progress bar, Default = TRUE
check logical, perform data check and attempt repair with check_df_visit(), computationally expensive on large data sets. Default: TRUE
env optional, provide environment of original visit data, Default: parent.frame()

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
simaerep object
See Also

site_aggr(), sim_sites(), eval_sites(), orivisit(), plot.simaerep()

Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with.ur = 0.4,
  ur_rate = 0.6
)

df_visit$study_id <- "A"

aerep <- simaerep(df_visit)

eaerep

str(aerep)
```

---

**sim_after_prep**

Start simulation after preparation.

Description

Internal function called by sim_sites after prep_for_sim

Usage

```r
sim_after_prep(
  df_sim_prep,
  r = 1000,
  poisson_test = FALSE,
  prob_lower = TRUE,
  progress = FALSE
)
```

Arguments

- `df_sim_prep` : dataframe as returned by prep_for_sim
- `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
**sim_scenario**

Value
dataframe

See Also

*sim_sites*, *prep_for_sim*

Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.2
)
df_visit$study_id <- "A"
df_site <- site_aggr(df_visit)
df_prep <- prep_for_sim(df_site, df_visit)
df_sim <- sim_after_prep(df_prep)
df_sim
```

---

**Description**

internal function called by simulate_scenarios()

**Usage**

```r
sim_scenario(n_ae_site, n_ae_study, frac_pat_with_ur, ur_rate)
```

**Arguments**

- `n_ae_site` integer vector
- `n_ae_study` integer vector
- `frac_pat_with_ur` double
- `ur_rate` double

**Value**

list
Examples

```
sim_scenario(c(5,5,5,5), c(8,8,8,8), 0.2, 0.5)
sim_scenario(c(5,5,5,5), c(8,8,8,8), 0.75, 0.5)
sim_scenario(c(5,5,5,5), c(8,8,8,8), 1, 0.5)
sim_scenario(c(5,5,5,5), c(8,8,8,8), 1, 1)
sim_scenario(c(5,5,5,5), c(8,8,8,8), 0, 0.5)
sim_scenario(c(5,5,5,5), c(8,8,8,8), 2, 0.5)
```

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.

Usage

```
sim_sites(
  df_site,  
  df_visit,  
  r = 1000,  
  poisson_test = TRUE,  
  prob_lower = TRUE,  
  progress = TRUE,  
  check = 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
- `check`: logical, perform data check and attempt repair with `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
sim_studies

Simulate studies.

Description

Test function, test applicability of poisson test, by calculating the bootstrapped probability of obtaining a specific p-value or lower, use in combination with get_ecd_values().

Usage

```r
sim_studies(
  df_visit,
  df_site,
  r = 100,
  poisson_test = TRUE,
)```

Examples

```r
df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.2
)

df_visit$study_id <- "A"

df_site <- site_aggr(df_visit)

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

df_sim_sites %>%
  knitr::kable(digits = 2)
```
prob_lower = TRUE,
  r_prob_lower = 1000,
  parallel = FALSE,
  keep_ae = FALSE,
  min_n_pat_with_med75 = 1,
  studies = NULL,
  .progress = TRUE
)

Arguments

  df_visit  dataframe
  df_site   dataframe
  r         integer, denotes number of simulations, Default: 1000
  poisson_test logical, calculates poisson.test pvalue, Default: TRUE
  prob_lower logical, calculates probability for getting a lower value, Default: FALSE
  r_prob_lower integer, denotes number of simulations for prob_lower value calculation, Default: 1000
  parallel  logical, see examples for registering parallel processing framework, Default: FALSE
  keep_ae   logical, keep ae numbers in output dataframe memory increase roughly 30 percent, Default: F
  min_n_pat_with_med75 integer, min number of patients with med75 at site to simulate, Default: 1
  studies   vector with study names, Default: NULL
  .progress logical, show progress bar

Details

Here we simulate study replicates maintaining the same number of sites, patients and visit_med75 by bootstrap resampling, then probabilities for obtaining lower or same mean_ae count and p-values using poisson.test are calculated.

adds column with simulated probabilities for equal or lower mean_ae at visit_med75

Value

dataframe

Examples

  df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 5,
                                frac_site_with_ur = 0.4, ur_rate = 0.6)
  df_visit1$study_id <- "A"
  df_visit2 <- sim_test_data_study(n_pat = 1000, n_sites = 3,
df_visit2$study_id <- "B"

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

df_site <- site_aggr(df_visit)

sim_studies(df_visit, df_site, r = 3, keep_ae = TRUE)

## Not run:
# parallel processing -------------------------
library(future)
future::plan(multiprocess)
sim_studies(df_visit, df_site, r = 3, keep_ae = TRUE, parallel = TRUE)
future::plan(sequential)

## End(Not run)

---

**sim_test_data_patient**  
*simulate patient ae reporting test data*

### Description

Helper function for `sim_test_data_study()`

### Usage

```r
sim_test_data_patient(
  .f_sample_max_visit = function() rnorm(1, mean = 20, sd = 4),
  .f_sample_ae_per_visit = function(max_visit) rpois(max_visit, 0.5)
)
```

### Arguments

- `.f_sample_max_visit`
  
  Function used to sample the maximum number of aEs, Default: `function() rnorm(1, mean = 20, sd = 4)`

- `.f_sample_ae_per_visit`
  
  Function used to sample the aEs for each visit, Default: `function(x) rpois(x, 0.5)`

### Details

```
```

### Value

Vector containing cumulative aEs
Examples

```r
replicate(5, sim_test_data_patient())
replicate(5, sim_test_data_patient(
  .f_sample_ae_per_visit = function(x) rpois(x, 1.2))
)
replicate(5, sim_test_data_patient(
  .f_sample_max_visit = function() rnorm(1, mean = 5, sd = 5))
)
```

---

**sim_test_data_portfolio**

*Simulate Portfolio Test Data*

Description

Simulate visit level data from a portfolio configuration.

Usage

```r
sim_test_data_portfolio(df_config, parallel = FALSE, progress = TRUE)
```

Arguments

- `df_config` dataframe as returned by `get_config`
- `parallel` logical activate parallel processing, see details, Default: FALSE
- `progress` logical, Default: TRUE

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
- **ae_per_visit_mean** mean AE per visit per study
- **site_number** site
- **max_visit_sd** standard deviation of maximum patient visits per site
- **max_visit_mean** mean of maximum patient visits per site
- **patnum** number of patients
- **visit** visit number
- **n_ae** cumulative sum of AEs
**Examples**

```r
df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10, 
frac_site_with_ur = 0.4, ur_rate = 0.6)
df_visit1$study_id <- "A"

df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10, 
frac_site_with_ur = 0.2, ur_rate = 0.1)
df_visit2$study_id <- "B"

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

df_site_max <- df_visit %>%
dplyr::group_by(study_id, site_number, patnum) %>%
dplyr::summarise(max_visit = max(visit), 
max_ae = max(n_ae),
.groups = "drop")

df_config <- get_config(df_site_max)

df_config
df_portf <- sim_test_data_portfolio(df_config)

df_portf
df_scen <- sim_ur_scenarios(df_portf, 
extra_ur_sites = 2, 
ur_rate = c(0.5, 1))

df_scen
df_perf <- get_portf_perf(df_scen)

df_perf
```
Description

evenly distributes a number of given patients across a number of given sites. Then simulates ae
development of each patient reducing the number of reported AEs for patients distributed to AE-
under-reporting sites.

Usage

`sim_test_data_study(
  n_pat = 1000,
  n_sites = 20,
  frac_site_with_ur = 0,
  ur_rate = 0,
  max_visit_mean = 20,
  max_visit_sd = 4,
  ae_per_visit_mean = 0.5
)
`

Arguments

- `n_pat` integer, number of patients, Default: 1000
- `n_sites` integer, number of sites, Default: 20
- `frac_site_with_ur` fraction of AE under-reporting sites, Default: 0
- `ur_rate` AE under-reporting rate, will lower mean ae per visit used to simulate patients
  at sites flagged as AE-under-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
- `ae_per_visit_mean` mean ae per visit per patient, Default: 0.5

Details

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

Value

tibble with columns site_number, patnum, is_ur, max_visit_mean, max_visit_sd, ae_per_visit_mean,
visit, n_ate

Examples

```r
set.seed(1)
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5)
df_visit[which(df_visit$patnum == "P000001"),]
df_visit <- sim_test_data_study(n_pat = 100, n_sites = 5,
    frac_site_with_ur = 0.2, ur_rate = 0.5)
df_visit[which(df_visit$patnum == "P000001"),]
```
Simulate Under-Reporting Scenarios

Description

Use with simulated portfolio data to generate under-reporting stats for specified scenarios.

Usage

```r
sim_ur_scenarios(
  df_portf,
  extra_ur_sites = 3,
  ur_rate = c(0.25, 0.5),
  r = 1000,
  poisson_test = FALSE,
  prob_lower = TRUE,
  parallel = FALSE,
  progress = TRUE,
  site_aggr_args = list(),
  eval_sites_args = list()
)
```

Arguments

- `df_portf` dataframe as returned by `sim_test_data_portfolio`
- `extra_ur_sites` numeric, set maximum number of additional under-reporting sites, see details Default: 3
- `ur_rate` numeric vector, set under-reporting rates for scenarios Default: c(0.25, 0.5)
- `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
- `parallel` logical, use parallel processing see details, Default: FALSE
- `progress` logical, show progress bar, Default: TRUE
- `site_aggr_args` named list of parameters passed to `site_aggr`, Default: list()
- `eval_sites_args` named list of parameters passed to `eval_sites`, Default: list()

Details

The function will apply under-reporting scenarios to each site. Reducing the number of AEs by a given under-reporting (ur_rate) for all patients at the site and add the corresponding under-reporting statistics. Since the under-reporting probability is also affected by the number of other sites that are under-reporting we additionally calculate under-reporting statistics in a scenario where additional under reporting sites are present. For this we use the median number of patients per site at the study
to calculate the final number of patients for which we lower the AEs in a given under-reporting scenario. 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 = 18)`

Value

dataframe with the following columns:

- **study_id**: study identification
- **site_number**: site identification
- **n_pat**: number of patients at site
- **n_pat_with_med75**: number of patients at site with visit_med75
- **visit_med75**: median(max(visit)) * 0.75
- **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 site with visit_med75 at study excl site
- **extra_ur_sites**: additional sites with under-reporting patients
- **frac_pat_with_ur**: ratio of patients in study that are under-reporting
- **ur_rate**: under-reporting rate
- **pval**: p-value as returned by `poisson.test`
- **prob_low**: bootstrapped probability for having mean_ae_site_med75 or lower
- **pval_adj**: adjusted p-values
- **prob_low_adj**: adjusted bootstrapped probability for having mean_ae_site_med75 or lower
- **pval_prob_ur**: probability under-reporting as 1 - pval_adj, poisson.test (use as benchmark)
- **prob_low_prob_ur**: probability under-reporting as 1 - prob_low_adj, bootstrapped (use)

See Also

- `sim_test_data_study`
- `get_config`
- `sim_test_data_portfolio`
- `sim_ur_scenarios`
- `get_portf_perf`

Examples

```r
df_visit1 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                       frac_site_with_ur = 0.4, ur_rate = 0.6)

df_visit1$study_id <- "A"

df_visit2 <- sim_test_data_study(n_pat = 100, n_sites = 10,
                       frac_site_with_ur = 0.2, ur_rate = 0.1)

df_visit2$study_id <- "B"

df_visit <- dplyr::bind_rows(df_visit1, df_visit2)
```
site_aggr <- df_visit %>%
dplyr::group_by(study_id, site_number, patnum) %>%
dplyr::summarise(max_visit = max(visit),
    max_ae = max(n_ae),
    .groups = "drop")

df_config <- get_config(df_site_max)

df_portf <- sim_test_data_portfolio(df_config)

df_scen <- sim_ur_scenarios(df_portf,
    extra_ur_sites = 2,
    ur_rate = c(0.5, 1))

df_perf <- get_portf_perf(df_scen)

---

site_aggr  

Aggregate from visit to site level.

Description

Calculates visit_med75, n_pat_with_med75 and mean_ae_site_med75

Usage

site_aggr(df_visit, method = "med75_adj", min_pat_pool = 0.2, check = TRUE)

Arguments

df_visit  dataframe with columns: study_id, site_number, patnum, visit, n_ae
method  character, one of c("med75", "med75_adj") 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
check,  logical, perform data check and attempt repair with check_df_visit(), computationally expensive on large data sets. Default: TRUE
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.

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

Examples

def df_visit <- sim_test_data_study(
  n_pat = 100,
  n_sites = 5,
  frac_site_with_ur = 0.4,
  ur_rate = 0.6
)
df_visit$study_id <- "A"
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

```r
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(
      ```
```r
rep(list(rep(0.25, 3)), 5), .purrr = purrr::walk, .f = f1, .steps = 5,
.f_args = list(progress = FALSE)
```