

Package ‘simtrial’

May 3, 2024

Type Package

Title Clinical Trial Simulation

Version 0.4.1

Description Provides some basic routines for simulating a clinical trial. The primary intent is to provide some tools to generate trial simulations for trials with time to event outcomes. Piecewise exponential failure rates and piecewise constant enrollment rates are the underlying mechanism used to simulate a broad range of scenarios such as those presented in Lin et al. (2020) <[doi:10.1080/19466315.2019.1697738](https://doi.org/10.1080/19466315.2019.1697738)>. However, the basic generation of data is done using pipes to allow maximum flexibility for users to meet different needs.

License GPL-3

URL <https://merck.github.io/simtrial/>,
<https://github.com/Merck/simtrial>

BugReports <https://github.com/Merck/simtrial/issues>

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports Rcpp, data.table (>= 1.12.4), doFuture, foreach, future, methods, mvtnorm, rlang, stats, survival, utils

Suggests Matrix, covr, dplyr, ggplot2, gsDesign, gsDesign2, gt, knitr, rmarkdown, survMisc, survRM2, testthat, tidy

LinkingTo Rcpp

RoxygenNote 7.3.1

NeedsCompilation yes

Author Keaven Anderson [aut],
Yujie Zhao [ctb, cre],
John Blischak [ctb],

Nan Xiao [ctb],
 Yilong Zhang [aut],
 Jianxiao Yang [ctb],
 Lili Ling [ctb],
 Xintong Li [ctb],
 Ruixue Wang [ctb],
 Yi Cui [ctb],
 Ping Yang [ctb],
 Yalin Zhu [ctb],
 Heng Zhou [ctb],
 Amin Shirazi [ctb],
 Cole Manschot [ctb],
 Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph]

Maintainer Yujie Zhao <yujie.zhao@merck.com>

Repository CRAN

Date/Publication 2024-05-03 15:40:02 UTC

R topics documented:

counting_process	3
create_cut	4
create_test	5
cut_data_by_date	6
cut_data_by_event	7
early_zero	7
ex1_delayed_effect	9
ex2_delayed_effect	10
ex3_cure_with_ph	11
ex4_belly	12
ex5_widening	13
ex6_crossing	14
fh	15
fit_pwexp	16
get_analysis_date	17
get_cut_date_by_event	22
maxcombo	23
mb	25
mb_delayed_effect	26
milestone	27
multitest	28
randomize_by_fixed_block	29
rmst	30
rpwexp	32
rpwexp_enroll	33
sim_fixed_n	34
sim_gs_n	37
sim_pw_surv	42

counting_process 3

to_sim_pw_surv 44
wlr 45

Index 48

counting_process *Process survival data into counting process format*

Description

Produces a data frame that is sorted by stratum and time. Included in this is only the times at which one or more event occurs. The output dataset contains stratum, TTE (time-to-event), at risk count, and count of events at the specified TTE sorted by stratum and TTE.

Usage

```
counting_process(x, arm)
```

Arguments

x A data frame with no missing values and contain variables:

- stratum: Stratum.
- treatment: Treatment group.
- tte: Observed time.
- event: Binary event indicator, 1 represents event, 0 represents censoring.

arm Value in the input treatment column that indicates treatment group value.

Details

The function only considered two group situation.

The tie is handled by the Breslow's Method.

The output produced by `counting_process()` produces a counting process dataset grouped by stratum and sorted within stratum by increasing times where events occur.

Value

A data frame grouped by stratum and sorted within stratum by tte. Remain rows with at least one event in the population, at least one subject is at risk in both treatment group and control group. Other variables in this represent the following within each stratum at each time at which one or more events are observed:

- events: Total number of events
- n_event_tol: Total number of events at treatment group
- n_risk_tol: Number of subjects at risk
- n_risk_trt: Number of subjects at risk in treatment group
- S: Left-continuous Kaplan-Meier survival estimate

- `o_minus_e`: In treatment group, observed number of events minus expected number of events. The expected number of events is estimated by assuming no treatment effect with hypergeometric distribution with parameters total number of events, total number of events at treatment group and number of events at a time. (Same assumption of log-rank test under the null hypothesis)
- `var_o_minus_e`: Variance of `o_minus_e` under the same assumption.

Examples

```
# Example 1
x <- data.frame(
  stratum = c(rep(1, 10), rep(2, 6)),
  treatment = rep(c(1, 1, 0, 0), 4),
  tte = 1:16,
  event = rep(c(0, 1), 8)
)
counting_process(x, arm = 1)

# Example 2
x <- sim_pw_surv(n = 400)
y <- cut_data_by_event(x, 150) |> counting_process(arm = "experimental")
# Weighted logrank test (Z-value and 1-sided p-value)
z <- sum(y$o_minus_e) / sqrt(sum(y$var_o_minus_e))
c(z, pnorm(z))
```

<code>create_cut</code>	<i>Create a cutting function</i>
-------------------------	----------------------------------

Description

Create a cutting function for use with `sim_gs_n()`

Usage

```
create_cut(...)
```

Arguments

... Arguments passed to `get_analysis_date()`

Value

A function that accepts a data frame of simulated trial data and returns a cut date

See Also

`get_analysis_date()`, `sim_gs_n()`

Examples

```
# Simulate trial data
trial_data <- sim_pw_surv()

# Create a cutting function that applies the following 2 conditions:
# - At least 45 months have passed since the start of the study
# - At least 300 events have occurred
cutting <- create_cut(
  planned_calendar_time = 45,
  target_event_overall = 350
)

# Cut the trial data
cutting(trial_data)
```

create_test

Create a cutting test function

Description

Create a cutting test function for use with [sim_gs_n\(\)](#)

Usage

```
create_test(test, ...)
```

Arguments

test	A test function such as wlr() , maxcombo() , or rmst()
...	Arguments passed to the cutting test function

Value

A function that accepts a data frame of simulated trial data and returns a test result

See Also

[sim_gs_n\(\)](#), [create_cut\(\)](#)

Examples

```
# Simulate trial data
trial_data <- sim_pw_surv()

# Cut after 150 events
trial_data_cut <- cut_data_by_event(trial_data, 150)

# Create a cutting test function that can be used by sim_gs_n()
regular_logrank_test <- create_test(wlr, weight = fh(rho = 0, gamma = 0))
```

```
# Test the cutting
regular_logrank_test(trial_data_cut)

# The results are the same as directly calling the function
stopifnot(all.equal(
  regular_logrank_test(trial_data_cut),
  wlr(trial_data_cut, weight = fh(rho = 0, gamma = 0))
))
```

cut_data_by_date	<i>Cut a dataset for analysis at a specified date</i>
------------------	---

Description

Cut a dataset for analysis at a specified date

Usage

```
cut_data_by_date(x, cut_date)
```

Arguments

x	A time-to-event dataset, for example, generated by sim_pw_surv() .
cut_date	Date relative to start of randomization (cte from input dataset) at which dataset is to be cut off for analysis.

Value

A dataset ready for survival analysis.

Examples

```
# Use default enrollment and event rates and
# cut at calendar time 5 after start of randomization
sim_pw_surv(n = 20) |> cut_data_by_date(5)
```

cut_data_by_event	<i>Cut a dataset for analysis at a specified event count</i>
-------------------	--

Description

Takes a time-to-event data set and cuts the data at which an event count is reached.

Usage

```
cut_data_by_event(x, event)
```

Arguments

x	A time-to-event dataset, for example, generated by <code>sim_pw_surv()</code> .
event	Event count at which data cutoff is to be made.

Value

A data frame ready for survival analysis, including columns time to event (tte), event, the stratum, and the treatment.

Examples

```
# Use default enrollment and event rates at cut at 100 events
x <- sim_pw_surv(n = 200) |> cut_data_by_event(100)
table(x$event, x$treatment)
```

early_zero	<i>Zero early weighting function</i>
------------	--------------------------------------

Description

Zero early weighting function

Usage

```
early_zero(early_period, fail_rate = NULL)
```

Arguments

early_period	The initial delay period where weights increase; after this, weights are constant at the final weight in the delay period.
fail_rate	Failure rate

Value

A list of parameters of the zero early weighting function

References

Xu, Z., Zhen, B., Park, Y., & Zhu, B. (2017). "Designing therapeutic cancer vaccine trials with delayed treatment effect."

Examples

```
library(gsDesign2)

# Example 1: Unstratified ----
sim_pw_surv(n = 200) |>
  cut_data_by_event(125) |>
  wlr(weight = early_zero(early_period = 2))

# Example 2: Stratified ----
n <- 500
# Two strata
stratum <- c("Biomarker-positive", "Biomarker-negative")
prevalence_ratio <- c(0.6, 0.4)

# Enrollment rate
enroll_rate <- define_enroll_rate(
  stratum = rep(stratum, each = 2),
  duration = c(2, 10, 2, 10),
  rate = c(c(1, 4) * prevalence_ratio[1], c(1, 4) * prevalence_ratio[2])
)
enroll_rate$rate <- enroll_rate$rate * n / sum(enroll_rate$duration * enroll_rate$rate)

# Failure rate
med_pos <- 10 # Median of the biomarker positive population
med_neg <- 8 # Median of the biomarker negative population
hr_pos <- c(1, 0.7) # Hazard ratio of the biomarker positive population
hr_neg <- c(1, 0.8) # Hazard ratio of the biomarker negative population
fail_rate <- define_fail_rate(
  stratum = rep(stratum, each = 2),
  duration = c(3, 1000, 4, 1000),
  fail_rate = c(log(2) / c(med_pos, med_pos, med_neg, med_neg)),
  hr = c(hr_pos, hr_neg),
  dropout_rate = 0.01
)

# Simulate data
temp <- to_sim_pw_surv(fail_rate) # Convert the failure rate
set.seed(2023)

sim_pw_surv(
  n = n, # Sample size
  # Stratified design with prevalence ratio of 6:4
```

```

stratum = data.frame(stratum = stratum, p = prevalence_ratio),
# Randomization ratio
block = c("control", "control", "experimental", "experimental"),
enroll_rate = enroll_rate, # Enrollment rate
fail_rate = temp$fail_rate, # Failure rate
dropout_rate = temp$dropout_rate # Dropout rate
) |>
cut_data_by_event(125) |>
wlr(weight = early_zero(early_period = 2, fail_rate = fail_rate))

```

ex1_delayed_effect	<i>Time-to-event data example 1 for non-proportional hazards working group</i>
--------------------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex1_delayed_effect)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuoka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex2_delayed_effect](#), [ex3_cure_with_ph](#), [ex4_belly](#), [ex5_widening](#), [ex6_crossing](#)

Examples

```
library(survival)

data(ex1_delayed_effect)
km1 <- with(ex1_delayed_effect, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
with(subset(ex1_delayed_effect, trt == 1), survfit(Surv(month, evntd) ~ trt))
with(subset(ex1_delayed_effect, trt == 0), survfit(Surv(month, evntd) ~ trt))
```

ex2_delayed_effect	<i>Time-to-event data example 2 for non-proportional hazards working group</i>
--------------------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex2_delayed_effect)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuoka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex1_delayed_effect](#), [ex3_cure_with_ph](#), [ex4_belly](#), [ex5_widening](#), [ex6_crossing](#)

Examples

```
library(survival)

data(ex2_delayed_effect)
km1 <- with(ex2_delayed_effect, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
with(subset(ex2_delayed_effect, trt == 1), survfit(Surv(month, evntd) ~ trt))
with(subset(ex2_delayed_effect, trt == 0), survfit(Surv(month, evntd) ~ trt))
```

ex3_cure_with_ph	<i>Time-to-event data example 3 for non-proportional hazards working group</i>
------------------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex3_cure_with_ph)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuoka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex1_delayed_effect](#), [ex2_delayed_effect](#), [ex4_belly](#), [ex5_widening](#), [ex6_crossing](#)

Examples

```
library(survival)

data(ex3_cure_with_ph)
km1 <- with(ex3_cure_with_ph, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
```

ex4_belly	<i>Time-to-event data example 4 for non-proportional hazards working group</i>
-----------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex4_belly)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex1_delayed_effect](#), [ex2_delayed_effect](#), [ex3_cure_with_ph](#), [ex5_widening](#), [ex6_crossing](#)

Examples

```
library(survival)

data(ex4_belly)
km1 <- with(ex4_belly, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
```

ex5_widening	<i>Time-to-event data example 5 for non-proportional hazards working group</i>
--------------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex5_widening)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuoka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex1_delayed_effect](#), [ex2_delayed_effect](#), [ex3_cure_with_ph](#), [ex4_belly](#), [ex6_crossing](#)

Examples

```
library(survival)

data(ex5_widening)
km1 <- with(ex5_widening, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
```

ex6_crossing	<i>Time-to-event data example 6 for non-proportional hazards working group</i>
--------------	--

Description

Survival objects reverse-engineered datasets from published Kaplan-Meier curves. Individual trials are de-identified since the data are only approximations of the actual data. Data are intended to evaluate methods and designs for trials where non-proportional hazards may be anticipated for outcome data.

Usage

```
data(ex6_crossing)
```

Format

Data frame with 4 variables:

- id: Sequential numbering of unique identifiers.
- month: Time-to-event.
- event: 1 for event, 0 for censored.
- trt: 1 for experimental, 0 for control.

References

Lin, Ray S., Ji Lin, Satrajit Roychoudhury, Keaven M. Anderson, Tianle Hu, Bo Huang, Larry F Leon, Jason J.Z. Liao, Rong Liu, Xiaodong Luo, Pralay Mukhopadhyay, Rui Qin, Kay Tatsuoka, Xuejing Wang, Yang Wang, Jian Zhu, Tai-Tsang Chen, Renee Iacona & Cross-Pharma Non-proportional Hazards Working Group. 2020. Alternative analysis methods for time to event endpoints under nonproportional hazards: A comparative analysis. *Statistics in Biopharmaceutical Research* 12(2): 187–198.

See Also

[ex1_delayed_effect](#), [ex2_delayed_effect](#), [ex3_cure_with_ph](#), [ex4_belly](#), [ex5_widening](#)

Examples

```
library(survival)

data(ex6_crossing)
km1 <- with(ex6_crossing, survfit(Surv(month, evntd) ~ trt))
km1
plot(km1)
```

fh *Fleming-Harrington weighting function*

Description

Fleming-Harrington weighting function

Usage

```
fh(rho = 0, gamma = 0)
```

Arguments

rho	Non-negative number. rho = 0, gamma = 0 is equivalent to regular logrank test.
gamma	Non-negative number. rho = 0, gamma = 0 is equivalent to regular logrank test.

Value

A list of parameters of the Fleming-Harrington weighting function

Examples

```
sim_pw_surv(n = 200) |>
  cut_data_by_event(100) |>
  wlr(weight = fh(rho = 0, gamma = 1))
```

fit_pwexp

*Piecewise exponential survival estimation***Description**

Computes survival function, density function, $-2 * \log$ -likelihood based on input dataset and intervals for piecewise constant failure rates. Initial version assumes observations are right censored or events only.

Usage

```
fit_pwexp(
  srv = Surv(time = ex1_delayed_effect$month, event = ex1_delayed_effect$evntd),
  intervals = array(3, 3)
)
```

Arguments

srv	Input survival object (see survival::Surv()); note that only 0 = censored, 1 = event for survival::Surv() .
intervals	Vector containing positive values indicating interval lengths where the exponential rates are assumed. Note that a final infinite interval is added if any events occur after the final interval specified.

Value

A matrix with rows containing interval length, estimated rate, $-2 * \log$ -likelihood for each interval.

Examples

```
# Use default arguments for delayed effect example dataset (ex1_delayed_effect)
library(survival)

# Example 1
rateall <- fit_pwexp()
rateall

# Example 2
# Estimate by treatment effect
rate1 <- with(subset(ex1_delayed_effect, trt == 1), fit_pwexp(Surv(month, evntd)))
rate0 <- with(subset(ex1_delayed_effect, trt == 0), fit_pwexp(Surv(month, evntd)))

rate1
rate0
rate1$rate / rate0$rate

# Chi-square test for (any) treatment effect (8 - 4 parameters = 4 df)
pchisq(sum(rateall$m2ll) - sum(rate1$m2ll + rate0$m2ll),
  df = 4,
```

```

    lower.tail = FALSE
  )

  # Compare with logrank
  survdiff(formula = Surv(month, evntd) ~ trt, data = ex1_delayed_effect)

  # Example 3
  # Simple model with 3 rates same for each for 3 months,
  # different for each treatment after months
  rate1a <- with(subset(ex1_delayed_effect, trt == 1), fit_pwexp(Surv(month, evntd), 3))
  rate0a <- with(subset(ex1_delayed_effect, trt == 0), fit_pwexp(Surv(month, evntd), 3))
  rate1a$rate / rate0a$rate

  m2110 <- rateall$m211[1] + rate1a$m211[2] + rate0a$m211[2]
  m2111 <- sum(rate0$m211) + sum(rate1$m211)

  # As a measure of strength, chi-square examines improvement in likelihood
  pchisq(m2110 - m2111, df = 5, lower.tail = FALSE)

```

get_analysis_date	<i>Derive analysis date for interim/final analysis given multiple conditions</i>
-------------------	--

Description

Derive analysis date for interim/final analysis given multiple conditions

Usage

```

get_analysis_date(
  data,
  planned_calendar_time = NA,
  target_event_overall = NA,
  target_event_per_stratum = NA,
  max_extension_for_target_event = NA,
  previous_analysis_date = 0,
  min_time_after_previous_analysis = NA,
  min_n_overall = NA,
  min_n_per_stratum = NA,
  min_followup = NA
)

```

Arguments

data A simulated data generated by `sim_pw_surv()`.

planned_calendar_time A numerical value specifying the planned calendar time for the analysis.

target_event_overall A numerical value specifying the targeted events for the overall population.

<code>target_event_per_stratum</code>	A numerical vector specifying the targeted events per stratum.
<code>max_extension_for_target_event</code>	A numerical value specifying the maximum time extension to reach targeted events.
<code>previous_analysis_date</code>	A numerical value specifying the previous analysis date.
<code>min_time_after_previous_analysis</code>	A numerical value specifying the planned minimum time after the previous analysis.
<code>min_n_overall</code>	A numerical value specifying the minimal overall sample size enrolled to kick off the analysis.
<code>min_n_per_stratum</code>	A numerical value specifying the minimal sample size enrolled per stratum to kick off the analysis.
<code>min_followup</code>	A numerical value specifying the minimal follow-up time after specified enrollment fraction in <code>min_n_overall</code> or <code>min_n_per_stratum</code> .

Details

To obtain the analysis date, consider the following multiple conditions:

Condition 1 The planned calendar time for analysis.

Condition 2 The targeted events, encompassing both overall population and stratum-specific events.

Condition 3 The maximum time extension required to achieve the targeted events.

Condition 4 The planned minimum time interval after the previous analysis.

Condition 5 The minimum follow-up time needed to reach a certain number of patients in enrollments.

Users have the flexibility to employ all 5 conditions simultaneously or selectively choose specific conditions to determine the analysis date. Any unused conditions will default to NA and not affect the output. Regardless of the number of conditions used, the analysis date is determined by $\min(\max(\text{date1}, \text{date2}, \text{date4}, \text{date5}, \text{na.rm} = \text{TRUE}), \text{date3}, \text{na.rm} = \text{TRUE})$, where `date1`, `date2`, `date3`, `date4`, `date5` represent the analysis dates determined solely by Condition 1, Condition 2, Condition 3, Condition 4 and Condition 5, respectively.

Value

A numerical value of the analysis date.

Examples

```
library(gsDesign2)

alpha <- 0.025
ratio <- 3
n <- 500
```

```

info_frac <- c(0.7, 1)
prevalence_ratio <- c(0.4, 0.6)
study_duration <- 48

# Two strata
stratum <- c("Biomarker-positive", "Biomarker-negative")

prevalence_ratio <- c(0.6, 0.4)
# enrollment rate
enroll_rate <- define_enroll_rate(
  stratum = rep(stratum, each = 2),
  duration = c(2, 10, 2, 10),
  rate = c(c(1, 4) * prevalence_ratio[1], c(1, 4) * prevalence_ratio[2])
)
enroll_rate$rate <- enroll_rate$rate * n / sum(enroll_rate$duration * enroll_rate$rate)

# Failure rate
med_pos <- 10 # Median of the biomarker positive population
med_neg <- 8 # Median of the biomarker negative population
hr_pos <- c(1, 0.7) # Hazard ratio of the biomarker positive population
hr_neg <- c(1, 0.8) # Hazard ratio of the biomarker negative population
fail_rate <- define_fail_rate(
  stratum = rep(stratum, each = 2),
  duration = 1000,
  fail_rate = c(log(2) / c(med_pos, med_pos, med_neg, med_neg)),
  hr = c(hr_pos, hr_neg),
  dropout_rate = 0.01
)

# Simulate data
temp <- to_sim_pw_surv(fail_rate) # Convert the failure rate
set.seed(2023)
simulated_data <- sim_pw_surv(
  n = n, # Sample size
  # Stratified design with prevalence ratio of 6:4
  stratum = data.frame(stratum = stratum, p = prevalence_ratio),
  # Randomization ratio
  block = c("control", "control", "experimental", "experimental"),
  enroll_rate = enroll_rate, # Enrollment rate
  fail_rate = temp$fail_rate, # Failure rate
  dropout_rate = temp$dropout_rate # Dropout rate
)

# Example 1: Cut for analysis at the 24th month.
# Here, we only utilize the `planned_calendar_time = 24` argument,
# while leaving the remaining unused arguments as their default value of `NA`.
get_analysis_date(
  simulated_data,
  planned_calendar_time = 24
)

# Example 2: Cut for analysis when there are 300 events in the overall population.
# Here, we only utilize the `target_event_overall = 300` argument,

```

```
# while leaving the remaining unused arguments as their default value of `NA`.
get_analysis_date(
  simulated_data,
  target_event_overall = 300
)

# Example 3: Cut for analysis at the 24th month and there are 300 events
# in the overall population, whichever arrives later.
# Here, we only utilize the `planned_calendar_time = 24` and
# `target_event_overall = 300` argument,
# while leaving the remaining unused arguments as their default value of `NA`.
get_analysis_date(
  simulated_data,
  planned_calendar_time = 24,
  target_event_overall = 300
)

# Example 4a: Cut for analysis when there are at least 100 events
# in the biomarker-positive population, and at least 200 events
# in the biomarker-negative population, whichever arrives later.
# Here, we only utilize the `target_event_per_stratum = c(100, 200)`,
# which refers to 100 events in the biomarker-positive population,
# and 200 events in the biomarker-negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by the number of events
# in each stratum.
get_analysis_date(
  simulated_data,
  target_event_per_stratum = c(100, 200)
)

# Example 4b: Cut for analysis when there are at least 100 events
# in the biomarker-positive population, but we don't have a requirement
# for the biomarker-negative population. Additionally, we want to cut
# the analysis when there are at least 150 events in total.
# Here, we only utilize the `target_event_overall = 150` and
# `target_event_per_stratum = c(100, NA)`, which refers to 100 events
# in the biomarker-positive population, and there is event requirement
# for the biomarker-negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by the number of events
# in the biomarker-positive population, and the total number of events,
# which arrives later.
get_analysis_date(
  simulated_data,
  target_event_overall = 150,
  target_event_per_stratum = c(100, NA)
)

# Example 4c: Cut for analysis when there are at least 100 events
# in the biomarker-positive population, but we don't have a requirement
# for the biomarker-negative population. Additionally, we want to cut
# the analysis when there are at least 150 events in total and after 24 months.
# Here, we only utilize the `planned_calendar_time = 24`,
# `target_event_overall = 150` and
```

```
# `target_event_per_stratum = c(100, NA)`, which refers to 100 events
# in the biomarker-positive population, and there is event requirement
# for the biomarker-negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by the number of events
# in the biomarker-positive population, the total number of events, and
# planned calendar time, which arrives later.
```

```
get_analysis_date(
  simulated_data,
  planned_calendar_time = 24,
  target_event_overall = 150,
  target_event_per_stratum = c(100, NA)
)
```

```
# Example 5: Cut for analysis when there are at least 100 events
# in the biomarker positive population, and at least 200 events
# in the biomarker negative population, whichever arrives later.
# But will stop at the 30th month if events are fewer than 100/200.
# Here, we only utilize the `max_extension_for_target_event = 30`,
# and `target_event_per_stratum = c(100, 200)`, which refers to
# 100/200 events in the biomarker-positive/negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by the number of events
# in the 2 strata, and the max extension to arrive at the targeted
# events, which arrives later.
```

```
get_analysis_date(
  simulated_data,
  target_event_per_stratum = c(100, 200),
  max_extension_for_target_event = 30
)
```

```
# Example 6a: Cut for analysis after 12 months followup when 80%
# of the patients are enrolled in the overall population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by
# 12 months + time when 80% patients enrolled.
```

```
get_analysis_date(
  simulated_data,
  min_n_overall = n * 0.8,
  min_followup = 12
)
```

```
# Example 6b: Cut for analysis after 12 months followup when 80%
# of the patients are enrolled in the overall population. Besides,
# the analysis happens when there are at least 150 events in total.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by the total number of events,
# and 12 months + time when 80% patients enrolled, which arrives later.
```

```
get_analysis_date(
  simulated_data,
  target_event_overall = 150,
  min_n_overall = n * 0.8,
  min_followup = 12
)
```

```

# Example 7a: Cut for analysis when 12 months after at least 200/160 patients
# are enrolled in the biomarker positive/negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by 12 months + time when there are
# 200/160 patients enrolled in the biomarker-positive/negative stratum.
get_analysis_date(
  simulated_data,
  min_n_per_stratum = c(200, 160),
  min_followup = 12
)
# Example 7b: Cut for analysis when 12 months after at least 200 patients
# are enrolled in the biomarker positive population, but we don't have a
# specific requirement for the biomarker negative population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by 12 months + time when there are
# 200 patients enrolled in the biomarker-positive stratum.
get_analysis_date(
  simulated_data,
  min_n_per_stratum = c(200, NA),
  min_followup = 12
)
# Example 7c: Cut for analysis when 12 months after at least 200 patients
# are enrolled in the biomarker-positive population, but we don't have a
# specific requirement for the biomarker-negative population. We also want
# there are at least 80% of the patients enrolled in the overall population.
# The remaining unused arguments as their default value of `NA`,
# so the analysis date is only decided by 12 months + max(time when there are
# 200 patients enrolled in the biomarker-positive stratum, time when there are
# 80% patients enrolled).
get_analysis_date(
  simulated_data,
  min_n_overall = n * 0.8,
  min_n_per_stratum = c(200, NA),
  min_followup = 12
)

```

get_cut_date_by_event *Get date at which an event count is reached*

Description

Get date at which an event count is reached

Usage

```
get_cut_date_by_event(x, event)
```

Arguments

`x` A time-to-event dataset, for example, generated by `sim_pw_surv()`.

`event` Event count at which dataset is to be cut off for analysis.

Value

A numeric value with the cte from the input dataset at which the targeted event count is reached, or if the final event count is never reached, the final cte at which an event occurs.

Examples

```
library(dplyr)

# Use default enrollment and calendar cut date
# for 50 events in the "Positive" stratum
x <- sim_pw_surv(
  n = 200,
  stratum = data.frame(
    stratum = c("Positive", "Negative"),
    p = c(.5, .5)
  ),
  fail_rate = data.frame(
    stratum = rep(c("Positive", "Negative"), 2),
    period = rep(1, 4),
    treatment = c(rep("control", 2), rep("experimental", 2)),
    duration = rep(1, 4),
    rate = log(2) / c(6, 9, 9, 12)
  ),
  dropout_rate = data.frame(
    stratum = rep(c("Positive", "Negative"), 2),
    period = rep(1, 4),
    treatment = c(rep("control", 2), rep("experimental", 2)),
    duration = rep(1, 4),
    rate = rep(.001, 4)
  )
)

d <- get_cut_date_by_event(x |> filter(stratum == "Positive"), event = 50)

y <- cut_data_by_date(x, cut_date = d)
table(y$stratum, y$event)
```

Description

WARNING: This experimental function is a work-in-progress. The function arguments will change as we add additional features.

Usage

```
maxcombo(
  data = cut_data_by_event(sim_pw_surv(n = 200), 150),
  rho = c(0, 0, 1),
  gamma = c(0, 1, 1),
  return_variance = FALSE,
  return_corr = FALSE
)
```

Arguments

<code>data</code>	A TTE dataset.
<code>rho</code>	Numeric vector. Must be greater than or equal to zero. Must be the same length as <code>gamma</code> .
<code>gamma</code>	Numeric vector. Must be greater than or equal to zero. Must be the same length as <code>rho</code> .
<code>return_variance</code>	A logical flag that, if TRUE, adds columns estimated variance for weighted sum of observed minus expected; see details; Default: FALSE.
<code>return_corr</code>	A logical flag that, if TRUE, adds columns estimated correlation for weighted sum of observed minus expected; see details; Default: FALSE.

Value

A list containing the test method (`method`), parameters of this test method (`parameter`), point estimation of the treatment effect (`estimation`), standardized error of the treatment effect (`se`), Z-score of each test of the MaxCombo (`z`), p-values (`p_value`) and the correlation matrix of each tests in MaxCombo (begin with `v`)

See Also

[wlr\(\)](#), [rmst\(\)](#), [milestone\(\)](#)

Examples

```
sim_pw_surv(n = 200) |>
  cut_data_by_event(150) |>
  maxcombo(rho = c(0, 0), gamma = c(0, 1), return_corr = TRUE)
```

mb *Magirr and Burman weighting function*

Description

Magirr and Burman weighting function

Usage

```
mb(delay = 4, w_max = Inf)
```

Arguments

delay	The initial delay period where weights increase; after this, weights are constant at the final weight in the delay period.
w_max	Maximum weight to be returned. Set delay = Inf, w_max = 2 to be consistent with recommendation of Magirr (2021).

Details

Magirr and Burman (2019) proposed a weighted logrank test to have better power than the logrank test when the treatment effect is delayed, but to still maintain good power under a proportional hazards assumption. In Magirr (2021), (the equivalent of) a maximum weight was proposed as opposed to a fixed time duration over which weights would increase. The weights for some early interval specified by the user are the inverse of the combined treatment group empirical survival distribution; see details. After this initial period, weights are constant at the maximum of the previous weights. Another advantage of the test is that under strong null hypothesis that the underlying survival in the control group is greater than or equal to underlying survival in the experimental group, Type I error is controlled as the specified level.

We define t^* to be the input variable delay. This specifies an initial period during which weights increase. We also set a maximum weight w_{\max} . To define specific weights, we let $S(t)$ denote the Kaplan-Meier survival estimate at time t for the combined data (control plus experimental treatment groups). The weight at time t is then defined as

$$w(t) = \min(w_{\max}, S(\min(t, t^*))^{-1}).$$

Value

A list of parameters of the Magirr and Burman weighting function

References

Magirr, Dominic, and Carl-Fredrik Burman. 2019. "Modestly weighted logrank tests." *Statistics in Medicine* 38 (20): 3782–3790.

Magirr, Dominic. 2021. "Non-proportional hazards in immuno-oncology: Is an old perspective needed?" *Pharmaceutical Statistics* 20 (3): 512–527.

Examples

```
sim_pw_surv(n = 200) |>
  cut_data_by_event(100) |>
  wlr(weight = mb(delay = 8, w_max = Inf))
```

mb_delayed_effect	<i>Simulated survival dataset with delayed treatment effect</i>
-------------------	---

Description

Magirr and Burman (2019) considered several scenarios for their modestly weighted logrank test. One of these had a delayed treatment effect with a hazard ratio of 1 for 6 months followed by a hazard ratio of 1/2 thereafter. The scenario enrolled 200 patients uniformly over 12 months and cut data for analysis 36 months after enrollment was opened. This dataset was generated by the `sim_pw_surv()` function under the above scenario.

Usage

```
mb_delayed_effect
```

Format

A data frame with 200 rows and 4 columns:

- tte: Time to event.

References

Magirr, Dominic, and Carl-Fredrik Burman. 2019. "Modestly weighted logrank tests." *Statistics in Medicine* 38 (20): 3782–3790.

Examples

```
library(survival)

fit <- survfit(Surv(tte, event) ~ treatment, data = mb_delayed_effect)

# Plot survival
plot(fit, lty = 1:2)
legend("topright", legend = c("control", "experimental"), lty = 1:2)

# Set up time, event, number of event dataset for testing
# with arbitrary weights
ten <- mb_delayed_effect |> counting_process(arm = "experimental")
head(ten)

# MaxCombo with logrank, FH(0,1), FH(1,1)
mb_delayed_effect |>
  maxcombo(rho = c(0, 0, 1), gamma = c(0, 1, 1), return_corr = TRUE)
```

```

# Generate another dataset
ds <- sim_pw_surv(
  n = 200,
  enroll_rate = data.frame(rate = 200 / 12, duration = 12),
  fail_rate = data.frame(
    stratum = c("All", "All", "All"),
    period = c(1, 1, 2),
    treatment = c("control", "experimental", "experimental"),
    duration = c(42, 6, 36),
    rate = c(log(2) / 15, log(2) / 15, log(2) / 15 * 0.6)
  ),
  dropout_rate = data.frame(
    stratum = c("All", "All"),
    period = c(1, 1),
    treatment = c("control", "experimental"),
    duration = c(42, 42),
    rate = c(0, 0)
  )
)
# Cut data at 24 months after final enrollment
mb_delayed_effect_2 <- ds |> cut_data_by_date(max(ds$enroll_time) + 24)

```

milestone

Milestone test for two survival curves

Description

Milestone test for two survival curves

Usage

```
milestone(data, ms_time, test_type = c("log-log", "naive"))
```

Arguments

data	Data frame containing at least 3 columns: <ul style="list-style-type: none"> • tte - Time to event. • event - Event indicator. • treatment - Grouping variable.
ms_time	Milestone analysis time.
test_type	Method to build the test statistics. There are 2 options: <ul style="list-style-type: none"> • "native": a native approach by dividing the KM survival difference by its standard derivations, see equation (1) of Klein, J. P., Logan, B., Harhoff, M., & Andersen, P. K. (2007). • "log-log": a log-log transformation of the survival, see equation (3) of Klein, J. P., Logan, B., Harhoff, M., & Andersen, P. K. (2007).

Value

A list frame containing:

- `method` - The method, always "milestone".
- `parameter` - Milestone time point.
- `estimation` - Survival difference between the experimental and control arm.
- `se` - Standard error of the control and experimental arm.
- `z` - Test statistics.

References

Klein, J. P., Logan, B., Harhoff, M., & Andersen, P. K. (2007). "Analyzing survival curves at a fixed point in time." *Statistics in Medicine*, 26(24), 4505–4519.

Examples

```
cut_data <- sim_pw_surv(n = 200) |>
  cut_data_by_event(150)

cut_data |>
  milestone(10, test_type = "log-log")

cut_data |>
  milestone(10, test_type = "naive")
```

multitest

Perform multiple tests on trial data cutting

Description

WARNING: This experimental function is a work-in-progress. The function arguments and/or returned output format may change as we add additional features.

Usage

```
multitest(data, ...)
```

Arguments

<code>data</code>	Trial data cut by <code>cut_data_by_event()</code> or <code>cut_data_by_date()</code>
<code>...</code>	One or more test functions. Use <code>create_test()</code> to change the default arguments of each test function.

Value

A list of test results, one per test. If the test functions are named in the call to `multitest()`, the returned list uses the same names.

See Also[create_test\(\)](#)**Examples**

```
trial_data <- sim_pw_surv(n = 200)
trial_data_cut <- cut_data_by_event(trial_data, 150)

# create cutting test functions
wlr_partial <- create_test(wlr, weight = fh(rho = 0, gamma = 0))
rmst_partial <- create_test(rmst, tau = 20)
maxcombo_partial <- create_test(maxcombo, rho = c(0, 0), gamma = c(0, 0.5))

multitest(
  data = trial_data_cut,
  wlr = wlr_partial,
  rmst = rmst_partial,
  maxcombo = maxcombo_partial
)
```

randomize_by_fixed_block

Permuted fixed block randomization

Description

Fixed block randomization. The block input should repeat each treatment code the number of times it is to be included within each block. The final block will be a partial block if n is not an exact multiple of the block length.

Usage

```
randomize_by_fixed_block(n = 10, block = c(0, 0, 1, 1))
```

Arguments

<code>n</code>	Sample size to be randomized.
<code>block</code>	Vector of treatments to be included in each block.

Value

A treatment group sequence (vector) of length n with treatments from `block` permuted within each block having block size equal to the length of `block`.

Examples

```
library(dplyr)

# Example 1
# 2:1 randomization with block size 3, treatments "A" and "B"
data.frame(x = 1:10) |> mutate(Treatment = randomize_by_fixed_block(block = c("A", "B", "B")))

# Example 2
# Stratified randomization
data.frame(stratum = c(rep("A", 10), rep("B", 10))) |>
  group_by(stratum) |>
  mutate(Treatment = randomize_by_fixed_block())
```

 rmst

RMST difference of 2 arms

Description

RMST difference of 2 arms

Usage

```
rmst(
  data,
  tau = 10,
  var_label_tte = "tte",
  var_label_event = "event",
  var_label_group = "treatment",
  formula = NULL,
  reference = "control",
  alpha = 0.05
)
```

Arguments

<code>data</code>	A time-to-event dataset with a column <code>tte</code> indicating the survival time and a column of event indicating whether it is event or censor.
<code>tau</code>	RMST analysis time.
<code>var_label_tte</code>	Column name of the TTE variable.
<code>var_label_event</code>	Column name of the event variable.
<code>var_label_group</code>	Column name of the grouping variable.

formula	(default: NULL) A formula that indicates the TTE, event, and group variables (in that exact order; see Details below). This is an alternative to specifying the variables as strings. If a formula is provided, the values passed to <code>var_label_tte</code> , <code>var_label_event</code> , and <code>var_label_group</code> are ignored.
reference	A group label indicating the reference group.
alpha	Type I error.

Details

The argument `formula` is provided as a convenience to easily specify the TTE, event, and grouping variables. Note however that only the order of the three variables is actually used by the underlying function. Any functions applied in the formula are ignored, and thus should only be used for documenting your intent. For example, you can use the syntax from the survival package `Surv(tte | event) ~ group` to highlight the relation between the TTE and event variables, but the function `Surv()` is never actually executed. Importantly, you shouldn't apply any transformation functions such as `log()` since these will also be ignored.

Value

The z statistics.

Examples

```
data(ex1_delayed_effect)
rmst(
  data = ex1_delayed_effect,
  var_label_tte = "month",
  var_label_event = "evntd",
  var_label_group = "trt",
  tau = 10,
  reference = "0"
)

# Formula interface
library("survival")

rmst(
  data = ex1_delayed_effect,
  formula = Surv(month | evntd) ~ trt,
  tau = 10,
  reference = "0"
)

# alternative
rmst(
  data = ex1_delayed_effect,
  formula = ~ Surv(month, evntd, trt),
  tau = 10,
  reference = "0"
)
```

```
# This example doesn't make statistical sense, but demonstrates that only the
# order of the 3 variables actually matters
rmst(
  data = ex1_delayed_effect,
  formula = month ~ evntd + trt,
  tau = 10,
  reference = "0"
)
```

rpwexp

The piecewise exponential distribution

Description

The piecewise exponential distribution allows a simple method to specify a distribution where the hazard rate changes over time. It is likely to be useful for conditions where failure rates change, but also for simulations where there may be a delayed treatment effect or a treatment effect that that is otherwise changing (for example, decreasing) over time. `rpwexp()` is to support simulation of both the Lachin and Foulkes (1986) sample size method for (fixed trial duration) as well as the Kim and Tsiatis (1990) method (fixed enrollment rates and either fixed enrollment duration or fixed minimum follow-up); see `gsDesign::nSurv()`.

Usage

```
rpwexp(n = 100, fail_rate = data.frame(duration = c(1, 1), rate = c(10, 20)))
```

Arguments

<code>n</code>	Number of observations to be generated.
<code>fail_rate</code>	A data frame containing duration and rate variables. <code>rate</code> specifies failure rates during the corresponding interval duration specified in <code>duration</code> . The final interval is extended to be infinite to ensure all observations are generated.

Details

Using the `cumulative = TRUE` option, enrollment times that piecewise constant over time can be generated.

Value

The generated random numbers.

Examples

```

# Example 1
# Exponential failure times
x <- rpwexp(
  n = 10000,
  fail_rate = data.frame(rate = 5, duration = 1)
)
plot(sort(x), (10000:1) / 10001,
     log = "y", main = "Exponential simulated survival curve",
     xlab = "Time", ylab = "P{Survival}")
)

# Example 2

# Get 10k piecewise exponential failure times.
# Failure rates are 1 for time 0 to 0.5, 3 for time 0.5 to 1, and 10 for > 1.
# Intervals specifies duration of each failure rate interval
# with the final interval running to infinity.
x <- rpwexp(
  n = 1e4,
  fail_rate = data.frame(rate = c(1, 3, 10), duration = c(.5, .5, 1))
)
plot(sort(x), (1e4:1) / 10001,
     log = "y", main = "PW Exponential simulated survival curve",
     xlab = "Time", ylab = "P{Survival}")
)

```

rpwexp_enroll

Generate piecewise exponential enrollment

Description

With piecewise exponential enrollment rate generation any enrollment rate distribution can be easily approximated. `rpwexp_enroll()` is to support simulation of both the Lachin and Foulkes (1986) sample size method for (fixed trial duration) as well as the Kim and Tsiatis(1990) method (fixed enrollment rates and either fixed enrollment duration or fixed minimum follow-up); see [gsDesign::nSurv\(\)](#).

Usage

```

rpwexp_enroll(
  n = NULL,
  enroll_rate = data.frame(duration = c(1, 2), rate = c(2, 5))
)

```

Arguments

`n` Number of observations. Default of NULL yields random enrollment size.

`enroll_rate` A data frame containing period duration (`duration`) and enrollment rate (`rate`) for specified enrollment periods. If necessary, last period will be extended to ensure enrollment of specified `n`.

Value

A vector of random enrollment times.

Examples

```
# Example 1
# Piecewise uniform (piecewise exponential inter-arrival times) for 10k patients enrollment
# Enrollment rates of 5 for time 0-100, 15 for 100-300, and 30 thereafter
x <- rpwexp_enroll(
  n = 1e5,
  enroll_rate = data.frame(
    rate = c(5, 15, 30),
    duration = c(100, 200, 100)
  )
)
plot(x, 1:1e5,
     main = "Piecewise uniform enrollment simulation",
     xlab = "Time",
     ylab = "Enrollment"
)

# Example 2
# Exponential enrollment
x <- rpwexp_enroll(
  n = 1e5,
  enroll_rate = data.frame(rate = .03, duration = 1)
)
plot(x, 1:1e5,
     main = "Simulated exponential inter-arrival times",
     xlab = "Time",
     ylab = "Enrollment"
)
```

sim_fixed_n

Simulation of fixed sample size design for time-to-event endpoint

Description

`sim_fixed_n()` provides simulations of a single endpoint two-arm trial where the enrollment, hazard ratio, and failure and dropout rates change over time.

Usage

```

sim_fixed_n(
  n_sim = 1000,
  sample_size = 500,
  target_event = 350,
  stratum = data.frame(stratum = "All", p = 1),
  enroll_rate = data.frame(duration = c(2, 2, 10), rate = c(3, 6, 9)),
  fail_rate = data.frame(stratum = "All", duration = c(3, 100), fail_rate = log(2)/c(9,
    18), hr = c(0.9, 0.6), dropout_rate = rep(0.001, 2)),
  total_duration = 30,
  block = rep(c("experimental", "control"), 2),
  timing_type = 1:5,
  rho_gamma = data.frame(rho = 0, gamma = 0)
)

```

Arguments

n_sim	Number of simulations to perform.
sample_size	Total sample size per simulation.
target_event	Targeted event count for analysis.
stratum	A data frame with stratum specified in stratum, probability (incidence) of each stratum in p.
enroll_rate	Piecewise constant enrollment rates by time period. Note that these are overall population enrollment rates and the stratum argument controls the random distribution between stratum.
fail_rate	Piecewise constant control group failure rates, hazard ratio for experimental vs. control, and dropout rates by stratum and time period.
total_duration	Total follow-up from start of enrollment to data cutoff.
block	As in sim_pw_surv() . Vector of treatments to be included in each block.
timing_type	A numeric vector determining data cutoffs used; see details. Default is to include all available cutoff methods.
rho_gamma	A data frame with variables rho and gamma, both greater than equal to zero, to specify one Fleming-Harrington weighted logrank test per row.

Details

timing_type has up to 5 elements indicating different options for data cutoff:

- 1: Uses the planned study duration.
- 2: The time the targeted event count is achieved.
- 3: The planned minimum follow-up after enrollment is complete.
- 4: The maximum of planned study duration and targeted event count cuts (1 and 2).
- 5: The maximum of targeted event count and minimum follow-up cuts (2 and 3).

Value

A data frame including columns:

- event: Event count.
- ln_hr: Log-hazard ratio.
- z: Normal test statistic; < 0 favors experimental.
- cut: Text describing cutoff used.
- duration: Duration of trial at cutoff for analysis.
- sim: Sequential simulation ID.

One row per simulated dataset per cutoff specified in `timing_type`, per test statistic specified. If multiple Fleming-Harrington tests are specified in `rho_gamma`, then columns `rho` and `gamma` are also included.

Examples

```
library(dplyr)
library(future)

# Example 1: logrank test ----
x <- sim_fixed_n(n_sim = 10, timing_type = 1, rho_gamma = data.frame(rho = 0, gamma = 0))
# Get power approximation
mean(x$z <= qnorm(.025))

# Example 2: WLR with FH(0,1) ----
sim_fixed_n(n_sim = 1, timing_type = 1, rho_gamma = data.frame(rho = 0, gamma = 1))
# Get power approximation
mean(x$z <= qnorm(.025))

# Example 3: MaxCombo, i.e., WLR-FH(0,0)+ WLR-FH(0,1)
# Power by test
# Only use cuts for events, events + min follow-up
x <- sim_fixed_n(
  n_sim = 10,
  timing_type = 2,
  rho_gamma = data.frame(rho = 0, gamma = c(0, 1))
)

# Get power approximation
x |>
  group_by(sim) |>
  filter(row_number() == 1) |>
  ungroup() |>
  summarize(power = mean(p_value < .025))

# Example 4
# Use two cores
set.seed(2023)
```

```
plan("multisession", workers = 2)
sim_fixed_n(n_sim = 10)
plan("sequential")
```

 sim_gs_n

Simulate group sequential designs with fixed sample size

Description

WARNING: This experimental function is a work-in-progress. The function arguments will change as we add additional features.

Usage

```
sim_gs_n(
  n_sim = 1000,
  sample_size = 500,
  stratum = data.frame(stratum = "All", p = 1),
  enroll_rate = data.frame(duration = c(2, 2, 10), rate = c(3, 6, 9)),
  fail_rate = data.frame(stratum = "All", duration = c(3, 100), fail_rate = log(2)/c(9,
    18), hr = c(0.9, 0.6), dropout_rate = rep(0.001, 2)),
  block = rep(c("experimental", "control"), 2),
  test = wlr,
  cut = NULL,
  seed = 2024,
  ...
)
```

Arguments

n_sim	Number of simulations to perform.
sample_size	Total sample size per simulation.
stratum	A data frame with stratum specified in stratum, probability (incidence) of each stratum in p.
enroll_rate	Piecewise constant enrollment rates by time period. Note that these are overall population enrollment rates and the stratum argument controls the random distribution between stratum.
fail_rate	Piecewise constant control group failure rates, hazard ratio for experimental vs. control, and dropout rates by stratum and time period.
block	As in <code>sim_pw_surv()</code> . Vector of treatments to be included in each block.
test	One or more test functions such as <code>wlr()</code> , <code>maxcombo()</code> , or <code>rmst()</code> . If a single test function is provided, it will be applied at each cut. Alternatively a list of functions created by <code>create_test()</code> . The list form is experimental and currently limited. It only accepts one test per cutting (in the future multiple tests

may be accepted), and all the tests must consistently return the same exact results (again this may be more flexible in the future). Importantly, note that the simulated data set is always passed as the first positional argument to each test function provided.

cut	A list of cutting functions created by <code>create_cut()</code> , see examples.
seed	Random seed.
...	Arguments passed to the test function(s) provided by the argument test.

Value

A data frame summarizing the simulation ID, analysis date, z statistics or p-values.

Examples

```
library(gsDesign2)

# Parameters for enrollment
enroll_rampup_duration <- 4 # Duration for enrollment ramp up
enroll_duration <- 16 # Total enrollment duration
enroll_rate <- define_enroll_rate(
  duration = c(
    enroll_rampup_duration,
    enroll_duration - enroll_rampup_duration
  ),
  rate = c(10, 30)
)

# Parameters for treatment effect
delay_effect_duration <- 3 # Delay treatment effect in months
median_ctrl <- 9 # Survival median of the control arm
median_exp <- c(9, 14) # Survival median of the experimental arm
dropout_rate <- 0.001
fail_rate <- define_fail_rate(
  duration = c(delay_effect_duration, 100),
  fail_rate = log(2) / median_ctrl,
  hr = median_ctrl / median_exp,
  dropout_rate = dropout_rate
)

# Other related parameters
alpha <- 0.025 # Type I error
beta <- 0.1 # Type II error
ratio <- 1 # Randomization ratio (experimental:control)

# Define cuttings of 2 IAs and 1 FA
# IA1
# The 1st interim analysis will occur at the later of the following 3 conditions:
# - At least 20 months have passed since the start of the study.
# - At least 100 events have occurred.
# - At least 20 months have elapsed after enrolling 200/400 subjects, with a
```

```
# minimum of 20 months follow-up.
# However, if events accumulation is slow, we will wait for a maximum of 24 months.
ia1 <- create_cut(
  planned_calendar_time = 20,
  target_event_overall = 100,
  max_extension_for_target_event = 24,
  min_n_overall = 200,
  min_followup = 20
)

# IA2
# The 2nd interim analysis will occur at the later of the following 3 conditions:
# - At least 32 months have passed since the start of the study.
# - At least 250 events have occurred.
# - At least 10 months after IA1.
# However, if events accumulation is slow, we will wait for a maximum of 34 months.
ia2 <- create_cut(
  planned_calendar_time = 32,
  target_event_overall = 200,
  max_extension_for_target_event = 34,
  min_time_after_previous_analysis = 10
)

# FA
# The final analysis will occur at the later of the following 2 conditions:
# - At least 45 months have passed since the start of the study.
# - At least 300 events have occurred.
fa <- create_cut(
  planned_calendar_time = 45,
  target_event_overall = 350
)

# Test 1: regular logrank test
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = wlr,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  weight = fh(rho = 0, gamma = 0)
)

# Test 2: weighted logrank test by FH(0, 0.5)
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = wlr,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
```

```
weight = fh(rho = 0, gamma = 0.5)
)

# Test 3: weighted logrank test by MB(3)
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = wlr,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  weight = mb(delay = 3)
)

# Test 4: weighted logrank test by early zero (6)
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = wlr,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  weight = early_zero(6)
)

# Test 5: RMST
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = rmst,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  tau = 20
)

# Test 6: Milestone
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = milestone,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  ms_time = 10
)

# Test 7: MaxCombo (WLR-FH(0,0) + WLR-FH(0, 0.5))
# for all analyses
```

```

sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = maxcombo,
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  rho = c(0, 0),
  gamma = c(0, 0.5)
)

# Test 8: MaxCombo (WLR-FH(0,0.5) + milestone(10))
# for all analyses
## Not run:
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = maxcombo(test1 = wlr, test2 = milestone),
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  test1_par = list(weight = fh(rho = 0, gamma = 0.5)),
  test2_par = list(ms_time = 10)
)

## End(Not run)

# Test 9: MaxCombo (WLR-FH(0,0) at IAs
# and WLR-FH(0,0) + milestone(10) + WLR-MB(4,2) at FA)
## Not run:
sim_gs_n(
  n_sim = 3,
  sample_size = 400,
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  test = list(ia1 = wlr, ia2 = wlr, fa = maxcombo),
  cut = list(ia1 = ia1, ia2 = ia2, fa = fa),
  seed = 2024,
  test_par = list(
    ia1 = list(weight = fh(rho = 0, gamma = 0)),
    ia2 = list(weight = fh(rho = 0, gamma = 0)),
    ia3 = list(
      test1_par = list(weight = fh(rho = 0, gamma = 0)),
      test2_par = list(ms_time = 10),
      test3_par = list(delay = 4, w_max = 2)
    )
  )
)

## End(Not run)

```

 sim_pw_surv

Simulate a stratified time-to-event outcome randomized trial

Description

sim_pw_surv() enables simulation of a clinical trial with essentially arbitrary patterns of enrollment, failure rates and censoring. The piecewise exponential distribution allows a simple method to specify a distribution and enrollment pattern where the enrollment, failure, and dropout rate changes over time. While the main purpose may be to generate a trial that can be analyzed at a single point in time or using group sequential methods, the routine can also be used to simulate an adaptive trial design. Enrollment, failure, and dropout rates are specified by treatment group, stratum and time period. Fixed block randomization is used; blocks must include treatments provided in failure and dropout specification. Default arguments are set up to allow very simple implementation of a non-proportional hazards assumption for an unstratified design.

Usage

```
sim_pw_surv(
  n = 100,
  stratum = data.frame(stratum = "All", p = 1),
  block = c(rep("control", 2), rep("experimental", 2)),
  enroll_rate = data.frame(rate = 9, duration = 1),
  fail_rate = data.frame(stratum = rep("All", 4), period = rep(1:2, 2), treatment =
    c(rep("control", 2), rep("experimental", 2)), duration = rep(c(3, 1), 2), rate =
    log(2)/c(9, 9, 9, 18)),
  dropout_rate = data.frame(stratum = rep("All", 2), period = rep(1, 2), treatment =
    c("control", "experimental"), duration = rep(100, 2), rate = rep(0.001, 2))
)
```

Arguments

n	Number of observations. If length(n) > 1, the length is taken to be the number required.
stratum	A data frame with stratum specified in stratum, probability (incidence) of each stratum in p.
block	Vector of treatments to be included in each block.
enroll_rate	Enrollment rates; see details and examples.
fail_rate	Failure rates; see details and examples; note that treatments need to be the same as input in block.
dropout_rate	Dropout rates; see details and examples; note that treatments need to be the same as input in block.

Value

A data frame with the following variables for each observation:

- stratum.
- enroll_time: Enrollment time for the observation.
- Treatment: Treatment group; this will be one of the values in the input block.
- fail_time: Failure time generated using `rpwexp()`.
- dropout_time: Dropout time generated using `rpwexp()`.
- cte: Calendar time of enrollment plot the minimum of failure time and dropout time.
- fail: Indicator that cte was set using failure time; i.e., 1 is a failure, 0 is a dropout.

Examples

```

library(dplyr)

# Example 1
sim_pw_surv(n = 20)

# Example 2
# 3:1 randomization
sim_pw_surv(
  n = 20,
  block = c(rep("experimental", 3), "control")
)

# Example 3
# Simulate 2 stratum; will use defaults for blocking and enrollRates
sim_pw_surv(
  n = 20,
  # 2 stratum, 30% and 70% prevalence
  stratum = data.frame(stratum = c("Low", "High"), p = c(.3, .7)),
  fail_rate = data.frame(
    stratum = c(rep("Low", 4), rep("High", 4)),
    period = rep(1:2, 4),
    treatment = rep(c(
      rep("control", 2),
      rep("experimental", 2)
    ), 2),
    duration = rep(c(3, 1), 4),
    rate = c(.03, .05, .03, .03, .05, .08, .07, .04)
  ),
  dropout_rate = data.frame(
    stratum = c(rep("Low", 2), rep("High", 2)),
    period = rep(1, 4),
    treatment = rep(c("control", "experimental"), 2),
    duration = rep(1, 4),
    rate = rep(.001, 4)
  )
)

# Example 4
# If you want a more rectangular entry for a data.frame
fail_rate <- bind_rows(

```

```

data.frame(stratum = "Low", period = 1, treatment = "control", duration = 3, rate = .03),
data.frame(stratum = "Low", period = 1, treatment = "experimental", duration = 3, rate = .03),
data.frame(stratum = "Low", period = 2, treatment = "experimental", duration = 3, rate = .02),
data.frame(stratum = "High", period = 1, treatment = "control", duration = 3, rate = .05),
data.frame(stratum = "High", period = 1, treatment = "experimental", duration = 3, rate = .06),
data.frame(stratum = "High", period = 2, treatment = "experimental", duration = 3, rate = .03)
)

dropout_rate <- bind_rows(
  data.frame(stratum = "Low", period = 1, treatment = "control", duration = 3, rate = .001),
  data.frame(stratum = "Low", period = 1, treatment = "experimental", duration = 3, rate = .001),
  data.frame(stratum = "High", period = 1, treatment = "control", duration = 3, rate = .001),
  data.frame(stratum = "High", period = 1, treatment = "experimental", duration = 3, rate = .001)
)

sim_pw_surv(
  n = 12,
  stratum = data.frame(stratum = c("Low", "High"), p = c(.3, .7)),
  fail_rate = fail_rate,
  dropout_rate = dropout_rate
)

```

to_sim_pw_surv	<i>Convert enrollment and failure rates from sim_fixed_n() to sim_pw_surv() format</i>
----------------	--

Description

to_sim_pw_surv() converts failure rates and dropout rates entered in the simpler format for [sim_fixed_n\(\)](#) to that used for [sim_pw_surv\(\)](#). The fail_rate argument for [sim_fixed_n\(\)](#) requires enrollment rates, failure rates hazard ratios and dropout rates by stratum for a 2-arm trial, [sim_pw_surv\(\)](#) is in a more flexible but less obvious but more flexible format. Since [sim_fixed_n\(\)](#) automatically analyzes data and [sim_pw_surv\(\)](#) just produces a simulation dataset, the latter provides additional options to analyze or otherwise evaluate individual simulations in ways that [sim_fixed_n\(\)](#) does not.

Usage

```

to_sim_pw_surv(
  fail_rate = data.frame(stratum = "All", duration = c(3, 100), fail_rate = log(2)/c(9,
    18), hr = c(0.9, 0.6), dropout_rate = rep(0.001, 2))
)

```

Arguments

fail_rate	Piecewise constant control group failure rates, hazard ratio for experimental vs. control, and dropout rates by stratum and time period.
-----------	--

Value

A list of two data frame components formatted for `sim_pw_surv()`: `fail_rate` and `dropout_rate`.

Examples

```
# Example 1
# Convert standard input
to_sim_pw_surv()

# Stratified example
fail_rate <- data.frame(
  stratum = c(rep("Low", 3), rep("High", 3)),
  duration = rep(c(4, 10, 100), 2),
  fail_rate = c(
    .04, .1, .06,
    .08, .16, .12
  ),
  hr = c(
    1.5, .5, 2 / 3,
    2, 10 / 16, 10 / 12
  ),
  dropout_rate = .01
)

x <- to_sim_pw_surv(fail_rate)

# Do a single simulation with the above rates
# Enroll 300 patients in ~12 months at constant rate
sim <- sim_pw_surv(
  n = 300,
  stratum = data.frame(stratum = c("Low", "High"), p = c(.6, .4)),
  enroll_rate = data.frame(duration = 12, rate = 300 / 12),
  fail_rate = x$fail_rate,
  dropout_rate = x$dropout_rate
)

# Cut after 200 events and do a stratified logrank test
sim |>
  cut_data_by_event(200) |> # Cut data
  wlr(weight = fh(rho = 0, gamma = 0)) # Stratified logrank
```

wlr

Weighted logrank test

Description

Weighted logrank test

Usage

```
wlr(data, weight, return_variance = FALSE)
```

Arguments

`data` Dataset that has been cut, generated by `sim_pw_surv()`.

`weight` Weighting functions, such as `fh()`, `mb()`, and `early_zero()`.

`return_variance` A logical flag that, if TRUE, adds columns estimated variance for weighted sum of observed minus expected; see details; Default: FALSE.

Details

- z - Standardized normal Fleming-Harrington weighted logrank test.
- i - Stratum index.
- d_i - Number of distinct times at which events occurred in stratum i .
- t_{ij} - Ordered times at which events in stratum i , $j = 1, 2, \dots, d_i$ were observed; for each observation, t_{ij} represents the time post study entry.
- O_{ij} - Total number of events in stratum i that occurred at time t_{ij} .
- O_{ije} - Total number of events in stratum i in the experimental treatment group that occurred at time t_{ij} .
- N_{ij} - Total number of study subjects in stratum i who were followed for at least duration.
- E_{ije} - Expected observations in experimental treatment group given random selection of O_{ij} from those in stratum i at risk at time t_{ij} .
- V_{ije} - Hypergeometric variance for E_{ije} as produced in Var from `counting_process()`.
- N_{ije} - Total number of study subjects in stratum i in the experimental treatment group who were followed for at least duration t_{ij} .
- E_{ije} - Expected observations in experimental group in stratum i at time t_{ij} conditioning on the overall number of events and at risk populations at that time and sampling at risk observations without replacement:

$$E_{ije} = O_{ij} \cdot N_{ije} / N_{ij}.$$

- S_{ij} - Kaplan-Meier estimate of survival in combined treatment groups immediately prior to time t_{ij} .
- ρ, γ - Real parameters for Fleming-Harrington test.
- X_i - Numerator for signed logrank test in stratum i

$$X_i = \sum_{j=1}^{d_i} S_{ij}^{\rho} (1 - S_{ij}^{\gamma}) (O_{ije} - E_{ije})$$

- V_{ij} - Variance used in denominator for Fleming-Harrington weighted logrank tests

$$V_i = \sum_{j=1}^{d_i} (S_{ij}^{\rho} (1 - S_{ij}^{\gamma}))^2 V_{ij}$$

The stratified Fleming-Harrington weighted logrank test is then computed as:

$$z = \sum_i X_i / \sqrt{\sum_i V_i}.$$

Value

A list containing the test method (`method`), parameters of this test method (`parameter`), point estimation of the treatment effect (`estimation`), standardized error of the treatment effect (`se`), Z-score (`z`), p-values (`p_value`).

Examples

```
x <- sim_pw_surv(n = 200) |> cut_data_by_event(100)

# Example 1: WLR test with FH wights
x |> wlr(weight = fh(rho = 0, gamma = 1))
x |> wlr(weight = fh(rho = 0, gamma = 1), return_variance = TRUE)

# Example 2: WLR test with MB wights
x |> wlr(weight = mb(delay = 4, w_max = 2))

# Example 3: WLR test with early zero wights
x |> wlr(weight = early_zero(early_period = 4))
```

Index

* datasets

- ex1_delayed_effect, 9
- ex2_delayed_effect, 10
- ex3_cure_with_ph, 11
- ex4_belly, 12
- ex5_widening, 13
- ex6_crossing, 14
- mb_delayed_effect, 26

counting_process, 3
counting_process(), 3, 46
create_cut, 4
create_cut(), 5, 38
create_test, 5
create_test(), 28, 29, 37
cut_data_by_date, 6
cut_data_by_date(), 28
cut_data_by_event, 7
cut_data_by_event(), 28

early_zero, 7
early_zero(), 46
ex1_delayed_effect, 9, 10–14
ex2_delayed_effect, 9, 10, 11–14
ex3_cure_with_ph, 9, 10, 11, 12–14
ex4_belly, 9–11, 12, 13, 14
ex5_widening, 9–12, 13, 14
ex6_crossing, 9–13, 14

fh, 15
fh(), 46
fit_pwexp, 16

get_analysis_date, 17
get_analysis_date(), 4
get_cut_date_by_event, 22
gsDesign::nSurv(), 32, 33

maxcombo, 23
maxcombo(), 5, 37
mb, 25

mb(), 46
mb_delayed_effect, 26
milestone, 27
milestone(), 24
multitest, 28

randomize_by_fixed_block, 29
rmst, 30
rmst(), 5, 24, 37
rpwexp, 32
rpwexp(), 43
rpwexp_enroll, 33

sim_fixed_n, 34
sim_fixed_n(), 44
sim_gs_n, 37
sim_gs_n(), 4, 5
sim_pw_surv, 42
sim_pw_surv(), 6, 7, 17, 23, 26, 35, 37, 44–46
survival::Surv(), 16

to_sim_pw_surv, 44

wlr, 45
wlr(), 5, 24, 37