Package 'simtrial'

December 11, 2023

Type Package

Title Clinical Trial Simulation

Version 0.3.2

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>. 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, doFuture, foreach, future, methods, mvtnorm, stats, survival, utils

Suggests Matrix, bshazard, covr, dplyr, ggplot2, gsDesign, gsDesign2, knitr, rmarkdown, survMisc, testthat, tidyr

LinkingTo Rcpp

RoxygenNote 7.2.3

NeedsCompilation yes

Author Keaven Anderson [aut], Yilong Zhang [aut], Yujie Zhao [ctb, cre],

Nan Xiao [ctb],
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],
John Blischak [ctb],
Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph]

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

Repository CRAN

Date/Publication 2023-12-11 15:30:02 UTC

R topics documented:

`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.

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))
```

`cut_data_by_date` *Cut a dataset for analysis at a specified date*

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 *Cut a dataset for analysis at a specified event count*

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 sim_pw_surv().
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_weight *Zero early weight for weighted logrank tests*

Description

Zero early weight for weighted logrank tests

Usage

```
early_zero_weight(x, early_period = 4, fail_rate = NULL)
```

Arguments

x	A counting_process()-class data frame with a counting process dataset.
early_period	The initial delay period where weights increase; after this, weights are constant at the final weight in the delay period.
fail_rate	A data frame record the failure rate.

Value

A data frame. The column `weight` contains the weights for the early zero weighted logrank test for the data in `x`.

References

Xu, Z., Zhen, B., Park, Y., & Zhu, B. (2017). "Designing therapeutic cancer vaccine trials with delayed treatment effect." *Statistics in Medicine*, 36(4), 592–605.

Examples

```
library(dplyr)
library(gsDesign2)

# Example 1: Unstratified
sim_pw_surv(n = 200) |>
  cut_data_by_event(125) |>
  counting_process(arm = "experimental") |>
  early_zero_weight(early_period = 2) |>
  filter(row_number() %in% seq(5, 200, 40))

# 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 = tibble(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) |>
  counting_process(arm = "experimental") |>
  early_zero_weight(early_period = 2, fail_rate = fail_rate) |>
  filter(row_number() %in% seq(5, 200, 40))
```

`ex1_delayed_effect` *Time-to-event data example 1 for non-proportional hazards working group*

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 *Time-to-event data example 2 for non-proportional hazards working group*

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 *Time-to-event data example 3 for non-proportional hazards working group*

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` *Time-to-event data example 4 for non-proportional hazards working group*

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 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, ex5_widening, ex6_crossing

Examples

```
library(survival)

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

`ex5_widening` *Time-to-event data example 5 for non-proportional hazards working group*

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` *Time-to-event data example 6 for non-proportional hazards working group*

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_weight` *Fleming-Harrington weighted logrank tests*

Description

With output from the function `counting_process()`.

Usage

```
fh_weight(
  x = counting_process(cut_data_by_event(sim_pw_surv(n = 200), 150), arm =
    "experimental"),
  rho_gamma = data.frame(rho = c(0, 0, 1, 1), gamma = c(0, 1, 0, 1)),
  return_variance = FALSE,
  return_corr = FALSE
)
```

Arguments

`x`	A `counting_process()`-class data frame with a counting process dataset.
`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; Default: `data.frame(rho = c(0, 0, 1, 1), gamma = c(0, 1, 0, 1))`.
`return_variance`	A logical flag that, if `TRUE`, adds columns estimated variance for weighted sum of observed minus expected; see details; Default: `FALSE`.
`return_corr`	A logical flag that, if `TRUE`, adds columns estimated correlation for weighted sum of observed minus expected; see details; Default: `FALSE`.

Details

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

- 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, $j = 1, 2, \ldots, d_i$ 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:

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

- 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 data frame with `rho_gamma` as input and the FH test statistic for the data in `x`. (`z`, a directional square root of the usual weighted logrank test); if variance calculations are specified (for example, to be used for covariances in a combination test), then this will be returned in the column `Var`.

Examples

```
library(dplyr)

# Example 1
# Use default enrollment and event rates at cut at 100 events
```

```
x <- sim_pw_surv(n = 200) |>
  cut_data_by_event(100) |>
  counting_process(arm = "experimental")

# Compute logrank FH(0, 1)
fh_weight(x, rho_gamma = data.frame(rho = 0, gamma = 1))
fh_weight(x, rho_gamma = data.frame(rho = 0, gamma = 1), return_variance = TRUE)

# Compute the corvariance between FH(0, 0), FH(0, 1) and FH(1, 0)
fh_weight(x, rho_gamma = data.frame(rho = c(0, 0, 1), gamma = c(0, 1, 0)))
fh_weight(x, rho_gamma = data.frame(rho = c(0, 0, 1), gamma = c(0, 1, 0)), return_variance = TRUE)
fh_weight(x, rho_gamma = data.frame(rho = c(0, 0, 1), gamma = c(0, 1, 0)), return_corr = TRUE)

# Example 2
# Use default enrollment and event rates at cut of 100 events
set.seed(123)
x <- sim_pw_surv(n = 200) |>
  cut_data_by_event(100) |>
  counting_process(arm = "experimental") |>
  fh_weight(rho_gamma = data.frame(rho = c(0, 0), gamma = c(0, 1)), return_corr = TRUE)

# Compute p-value for MaxCombo
library(mvtnorm)
1 - pmvnorm(
  lower = rep(min(x$z), nrow(x)),
  corr = data.matrix(select(x, -c(rho, gamma, z))),
  algorithm = GenzBretz(maxpts = 50000, abseps = 0.00001)
)[1]

# Check that covariance is as expected
x <- sim_pw_surv(n = 200) |>
  cut_data_by_event(100) |>
  counting_process(arm = "experimental")

x |> fh_weight(
  rho_gamma = data.frame(
    rho = c(0, 0),
    gamma = c(0, 1)
  ),
  return_variance = TRUE
)

# Off-diagonal element should be variance in following
x |> fh_weight(
  rho_gamma = data.frame(
    rho = 0,
    gamma = .5
  ),
  return_variance = TRUE
)

# Compare off diagonal result with fh_weight()
x |> fh_weight(rho_gamma = data.frame(rho = 0, gamma = .5))
```

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

m2ll0 <- rateall$m2ll[1] + rate1a$m2ll[2] + rate0a$m2ll[2]
m2ll1 <- sum(rate0$m2ll) + sum(rate1$m2ll)

# As a measure of strength, chi-square examines improvement in likelihood
pchisq(m2ll0 - m2ll1, df = 5, lower.tail = FALSE)
```

`get_analysis_date` *Get the analysis date under multiple conditions*

Description

Get the analysis date under 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.

target_event_per_stratum
: A numerical vector specifying the targeted events per stratum.

max_extension_for_target_event
: A numerical value specifying the maximum time extension to reach targeted events.

previous_analysis_date
: A numerical value specifying the previous analysis date.

min_time_after_previous_analysis
: A numerical value specifying the planned minimum time after the previous analysis.

min_n_overall
: A numerical value specifying the minimal overall sample size enrolled to kick off the analysis.

min_n_per_stratum
: A numerical value specifying the minimal sample size enrolled per stratum to kick off the analysis.

min_followup
: A numerical value specifying the minimal follow-up time after specified enrollment fraction in `min_n_overall` or `min_n_per_stratum`.

Value

A numerical value of the analysis date.

Examples

```
library(gsDesign2)
library(simtrial)

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.
get_analysis_date(
  simulated_data,
  planned_calendar_time = 24
)

# Example 2: Cut for analysis when there are 300 events in the overall population.
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.
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.
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.
```

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

# Example 6: Cut for analysis after 12 months followup when 80%
# of the patients are enrolled in the overall population.
get_analysis_date(
  simulated_data,
  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.
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.
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.
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)
```

mb_delayed_effect *Simulated survival dataset with delayed treatment effect*

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)
library(dplyr)

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)
ten |>
 fh_weight(rho_gamma = data.frame(rho = c(0, 0, 1), gamma = c(0, 1, 1)), return_corr = TRUE) |>
  pvalue_maxcombo()

# Magirr-Burman modestly down-weighted rank test with 6 month delay
# First, add weights
```

```
ten <- ten |> mb_weight(6)
head(ten)

# Now compute test based on these weights
ten |>
  summarize(
    S = sum(o_minus_e * mb_weight),
    V = sum(var_o_minus_e * mb_weight^2),
    Z = S / sqrt(V)
  ) |>
  mutate(p = pnorm(Z))

# Create 0 weights for first 6 months
ten <- ten |> mutate(w6 = 1 * (tte >= 6))
ten |>
  summarize(
    S = sum(o_minus_e * w6),
    V = sum(var_o_minus_e * w6^2),
    Z = S / sqrt(V)
  ) |>
  mutate(p = pnorm(Z))

# 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)
```

---

mb_weight *Magirr and Burman modestly weighted logrank tests*

Description

Computes Magirr-Burman weights and adds them to a dataset created by counting_process(). These weights can then be used to compute a z-statistic for the modestly weighted logrank test

proposed.

Usage

```
mb_weight(x, delay = 4, w_max = Inf)
```

Arguments

x	A counting_process()-class data frame with a counting process dataset.
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 data frame. The column `mb_weight` contains the weights for the Magirr-Burman weighted logrank test for the data in `x`.

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

```
library(dplyr)

# Use default enrollment and event rates at cut at 100 events
```

```
# For transparency, may be good to set either `delay` or `w_max` to `Inf`
x <- sim_pw_surv(n = 200) |>
  cut_data_by_event(125) |>
  counting_process(arm = "experimental")

# Example 1
# Compute Magirr-Burman weights with `delay = 6`
ZMB <- x |>
  mb_weight(delay = 6, w_max = Inf) |>
  summarize(
    S = sum(o_minus_e * mb_weight),
    V = sum(var_o_minus_e * mb_weight^2),
    z = S / sqrt(V)
  )

# Compute p-value of modestly weighted logrank of Magirr-Burman
pnorm(ZMB$z)

# Example 2
# Now compute with maximum weight of 2 as recommended in Magirr, 2021
ZMB2 <- x |>
  mb_weight(delay = Inf, w_max = 2) |>
  summarize(
    S = sum(o_minus_e * mb_weight),
    V = sum(var_o_minus_e * mb_weight^2),
    z = S / sqrt(V)
  )

# Compute p-value of modestly weighted logrank of Magirr-Burman
pnorm(ZMB2$z)
```

`pvalue_maxcombo` *MaxCombo p-value*

Description

Computes p-values for the MaxCombo test based on output from `fh_weight()`. This is still in an experimental stage and is intended for use with the `sim_fixed_n()` trial simulation routine. However, it can also be used to analyze clinical trial data such as that provided in the ADaM ADTTE format.

Usage

```
pvalue_maxcombo(
  z,
  algorithm = mvtnorm::GenzBretz(maxpts = 50000, abseps = 1e-05)
)
```

Arguments

z	A dataset output from fh_weight(); see examples.
algorithm	This is passed directly to the algorithm argument in mvtnorm::pmvnorm().

Value

A numeric p-value.

Examples

```
library(dplyr)

# Example 1
x <- sim_fixed_n(
  n_sim = 1,
  timing_type = 5,
  rho_gamma = data.frame(
    rho = c(0, 0, 1),
    gamma = c(0, 1, 1)
  )
)
head(x)
pvalue_maxcombo(x)

# Example 2
# Only use cuts for events, events + min follow-up
xx <- sim_fixed_n(
  n_sim = 100,
  timing_type = 5,
  rho_gamma = data.frame(
    rho = c(0, 0, 1),
    gamma = c(0, 1, 1)
  )
)
head(xx)

# MaxCombo power estimate for cutoff at max of targeted events, minimum follow-up
p <- xx |>
  group_by(sim) |>
  group_map(~ pvalue_maxcombo(.x)) |>
  unlist()
mean(p < .025)
```

---

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

`n`	Sample size to be randomized.
`block`	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())
```

---

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

n	Number of observations to be generated.
fail_rate	A data frame containing duration and rate variables. rate specifies failure rates during the corresponding interval duration specified in duration. 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 As in fh_weight(). 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
# Show output structure
sim_fixed_n(n_sim = 3)

# Example 2
# Example with 2 tests: logrank and FH(0,1)
sim_fixed_n(n_sim = 1, rho_gamma = data.frame(rho = 0, gamma = c(0, 1)))


# Example 3
# Power by test
# Only use cuts for events, events + min follow-up
xx <- sim_fixed_n(
  n_sim = 100,
  timing_type = c(2, 5),
```

```
  rho_gamma = data.frame(rho = 0, gamma = c(0, 1))
)
# Get power approximation for FH, data cutoff combination
xx |>
  group_by(cut, rho, gamma) |>
  summarize(mean(z <= qnorm(.025)))

# MaxCombo power estimate for cutoff at max of targeted events, minimum follow-up
p <- xx |>
  filter(cut != "Targeted events") |>
  group_by(sim) |>
  group_map(~ pvalue_maxcombo(.x)) |>
  unlist()

mean(p < .025)

# MaxCombo estimate for targeted events cutoff
p <- xx |>
  filter(cut == "Targeted events") |>
  group_by(sim) |>
  group_map(~ pvalue_maxcombo(.x)) |>
  unlist()

mean(p < .025)

# Example 4
# Use two cores
set.seed(2023)
plan("multisession", workers = 2)
sim_fixed_n(n_sim = 10)
plan("sequential")
```

`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` *Convert enrollment and failure rates from* `sim_fixed_n()` *to* `sim_pw_surv()` *format*

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
dat <- sim |>
  cut_data_by_event(200) |> # Cut data
  counting_process(arm = "experimental") |> # Convert format for fh_weight()
  fh_weight(rho_gamma = data.frame(rho = 0, gamma = 0)) # Stratified logrank
```

Index