Package 'gsMeanFreq'

February 16, 2026

Type Package

Title Group Sequential Clinical Trial Designs for Composite Endpoints

Version 0.1.0

Maintainer Qinghua Lian <qlian@mcw.edu>

Description Simulating composite endpoints with recurrent and terminal events under staggered entry, and for constructing one- and two-sample group sequential test statistics and monitoring boundaries based on the mean frequency function. Details will be available in an upcoming publication.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, tibble, pracma, mvtnorm, gsDesign, stats, bdsmatrix, rlang, foreach, survival

NeedsCompilation no

Author Qinghua Lian [aut, cre],
Kwang Woo Ahn [ctb],
Soyoung Kim [ctb],
Michael J Martens [aut],
Brent R. Logan [ctb]

Repository CRAN

Date/Publication 2026-02-16 17:50:02 UTC

Contents

`Apply.calendar.censoring`

Function to apply a given calendar time as effective censoring time.

Description

Applies administrative censoring at a specified calendar time by truncating each subject's observed event history. For subjects who are still under observation at the calendar cutoff, a censoring record is added at the specified calendar time.

Usage

```
Apply.calendar.censoring(data, calendar)
```

Arguments

`data`	A data frame containing simulated composite endpoint data generated by `Onesample.generate.sequent`
`calendar`	A positive numeric value specifying the calendar time (in years) at which administrative censoring is applied.

Value

A data frame in long format containing the censored composite endpoint data. Each subject contributes all events occurring on or before the specified calendar time, with an additional censoring record added for subjects who have not experienced a terminal event by that time.

Examples

```
df <- Onesample.generate.sequential(size = 200, recruitment = 3,
calendar = 5, random.censor.rate = 0.05, seed = 1123)
df2 <- Apply.calendar.censoring(data = df, calendar = 4)
```

`Apply.calendar.censoring.2` *Function to apply a given calendar time as effective censoring time for two-sample composite endpoint data.*

Description

Applies administrative censoring at a specified calendar time by truncating each subject's observed event history; for subjects still under observation at the calendar cutoff, a censoring record is added at the cutoff time.

Usage

```
Apply.calendar.censoring.2(data, calendar)
```

Arguments

`data`	A data frame generated by `TwoSample.generate.sequential()` containing simulated two-sample composite endpoint data in long format.
`calendar`	A positive numeric value specifying the calendar time (in years) at which administrative censoring is applied.

Value

A data frame in long format containing the censored composite endpoint data. Each subject contributes all events occurring on or before the specified calendar time, with an additional censoring record added for subjects who have not experienced a terminal event by that time.

Examples

```
df <- TwoSample.generate.sequential(sizevec = c(100, 100), beta.trt = 0.8,
calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
df2 <- Apply.calendar.censoring.2(data= df, calendar = 3.9)
```

`find.beta.trt` *Function to solve for treatment effect size given target power for two-sample tests.*

Description

Computes the treatment effect parameter `beta.trt` that achieves a specified power for a two-sided Wald-type test under a normal approximation. The calculation assumes that the test statistic Q is asymptotically normal with mean proportional to the treatment effect and variance `var`.

Usage

```
find.beta.trt(power = 0.8, power.c, var)
```

Arguments

power	Target power for the two-sided test, default is 0.8.
power.c	Positive numeric constant relating the mean of the test statistic under the alternative hypothesis to the treatment effect.
var	Variance of the test statistic Q under the alternative hypothesis.

Value

A numeric value giving the treatment effect parameter.

---

find.c *Function to solve for a group sequential critical value at a given stage*

Description

Computes the critical boundary value at the current analysis stage by solving a probability equation under a multivariate normal distribution. The boundary is chosen so that the incremental Type I error spent at the current stage equals a specified target value, conditional on previously determined boundaries.

Usage

```
find.c(previous.c, pi, corr)
```

Arguments

previous.c	Numeric value giving the critical boundary from the previous analysis stage.
pi	Target incremental Type I error to be spent at the current stage.
corr	Correlation matrix of the joint multivariate normal distribution of test statistics across analyses.

Value

A numeric value giving the critical boundary c at the current analysis stage that satisfies the specified Type I spending constraint.

`find.Delta.given.power` *Function to solve for the mean frequency effect size given target power for one-sample simulations.*

Description

Computes the mean frequency difference parameter Δ that achieves a specified power for a two-sided group sequential test.

Usage

```
find.Delta.given.power(K, timing, alpha, power, Imax)
```

Arguments

`K`	Number of analysis (including the final analysis).
`timing`	Numeric vector of information fractions at each analysis, with values in $(0, 1)$ and length equal to `K`.
`alpha`	Overall Type I error.
`power`	Target power of thw group sequenctial test.
`Imax`	Maximum information at the final analysis.

Value

A numeric value giving the mean frequency effect size Δ that achieves the specified power.

Examples

```
# Toy Example: n = 800, two interim analysi planned at 50% and
# 75% if maximum information, along with a final analysis
# at 100% maximum information. Assuming the maximum
# information is 100, overall Type I error at 0.05 and power at 0.8.
find.Delta.given.power(K = 3, timing = c(0.5, 0.75, 1),
 alpha = 0.05, power = 0.8, Imax = 100)
```

`find.lambda_0.given.Delta` *Function to solve for the baseline recurrent event rate given a mean frequency difference for one-sample simulations.*

Description

Computes the value of the baseline recurrent event rate parameter λ_0 hat yields a specified difference in mean frequency function at a given event time, relative to a null value.

Usage

```
find.lambda_0.given.Delta(
  lambda_star,
  null.lambda_0,
  gamma_shape,
  gamma_scale,
  Delta,
  t
)
```

Arguments

lambda_star	Rate parameter of an exponential distribution in generating the terminal event.
null.lambda_0	Baseline recurrent event rate under the null hypothesis.
gamma_shape	Shape parameter of the Gamma frailty distribution.
gamma_scale	Scale parameter of the Gamma frailty distribution.
Delta	Target difference in the mean frequency function at time t between the null and alternative.
t	Event time (time since enrollment) at which the mean frequency difference is defined.

Value

A numeric value giving the baseline recurrent event rate λ_0 that satisfies the specified mean frequency difference at time t.

Examples

```
# Toy Example: n = 800, two interim analysi planned
# at 50% and 75% if maximum information, along with
# a final analysis at 100% maximum information.
# Assuming the maximum information is 100,
# overall Type I error at 0.05 and power at 0.8.
find.Delta.given.power(K = 3, timing = c(0.5, 0.75, 1),
alpha = 0.05, power = 0.8, Imax = 100)
# Use the computed value from the
# "find.Delta.given.power()" function
find.lambda_0.given.Delta(lambda_star = 0.1, null.lambda_0 = 1.15,
gamma_shape = 2, gamma_scale = 0.5, Delta = 0.2834591, t = 2)
```

find.lambda_0.given.mu0

Function to solve for the baseline recurrent event rate given a target mean frequency for one-sample simulations.

Description

Computes the value of the baseline recurrent event rate parameter.

Usage

```
find.lambda_0.given.mu0(lambda_star, gamma_shape, gamma_scale, t, mu0)
```

Arguments

lambda_star	Rate parameter of an exponential distribution in generating the terminal event.
gamma_shape	Shape parameter of the Gamma frailty distribution.
gamma_scale	Scale parameter of the Gamma frailty distribution.
t	Event time (time since enrollment) at which the mean frequency value mu0 is defined.
mu0	Target value of the mean frequency function at time t.

Value

A numeric value giving the baseline recurrent event rate λ_0 that satisfies $\mu(t) = \mu(0)$.

Examples

```
find.lambda_0.given.mu0(lambda_star = 0.1, gamma_shape = 2, gamma_scale = 0.5, t = 1, mu0 = 2)
```

OBF

O'Brien-Flemming error spending function.

Description

Computes the cumulative Type I error spending function corresponding to the two-sided O'Brien-Flemming group sequential design.

Usage

```
OBF(t, alpha = 0.05)
```

Arguments

t	Mumeric vector of information fractions, typically in $(0, 1)$, where t=1 corresponds to the final analysis.
alpha	Overall two-sided Type I error level. Default is 0.05.

Value

A numeric vector giving the cumulative type I error spent at each information fraction t under the O'Brien-Flemming spending function.

References

Jennison, C. and Turnbull, B. W. (2000). *Group Sequential Methods with Applications to Clinical Trials*. Chapman and Hall/CRC.

Examples

```
# Two-interim analyses at 50% and 75% maximum information and a final analysis
OBF(t = c(0.5, 0.7, 1))
```

OneSample.Estimator.sequential

Function to estimate the one-sample mean frequency under a group sequential design.

Description

Computes the nonparametric one-sample estimator of the mean frequency function for composite endpoints consisting of recurrent events and terminal event (death), using data observed up to a given calendar time. The estimator is constructed on the event-time scale (time since subject enrollment).

Usage

```
OneSample.Estimator.sequential(data, t = NULL)
```

Arguments

data	A data frame generated by Onesample.generate.sequential() (optionally after applying Apply.calendar.censoring()) containing composite endpoint data in long format. Each subject may contribute multiple rows corresponding to recurrent events, terminal event, or censoring.
t	Optional numeric vector specifying event times (time since enrollment) at which the mean frequency estimator and its variance are evaluated. Default is NULL.

Value

If `t` is provided, returns a list containing:

- `muhat`: Estimated mean frequency function evaluated at `t`.
- `varhat`: Estimated variance of `muhat` at `t`.
- `n`: Sample size of the input dataset.

If `t` is `NULL`, returns a numeric vector `xihat` giving the estimated cumulative variance component evaluated at all ordered observed event time. This quantity is not the mean frequency estimator itself. @importFrom dplyr group_by filter mutate count select slice ungroup @importFrom tibble as_tibble @importFrom bdsmatrix bdsBlock @importFrom rlang .data

Examples

```
df <- Onesample.generate.sequential(size = 400, recruitment = 3,
calendar = 5, random.censor.rate = 0.05, seed = 1123)
OneSample.Estimator.sequential(data= df, t = c(1.5, 2.9, 4.6))
```

Onesample.generate.sequential

Function to simulate one-sample composite endpoint data under staggered entry.

Description

Simulate one-sample composite endpoints data with recurrent events and a terminal event under two time scales: event time `t` and calendar time `s`. A uniform recruitment period is assumed, and the function returns all observed data available at a specified calendar time. Recurrent event occurrences are generated from an underlying Poisson process with subject-specific Gamma frailty.

Usage

```
Onesample.generate.sequential(
  beta = c(0, 0),
  lambda_0 = 1.15,
  size,
  recruitment = 3,
  calendar = 5,
  random.censor.rate,
  seed
)
```

Arguments

beta	Regression coefficient vector for the proportional mean functions, the default is c(0,0), corresponds to no covariates effects.
lambda_0	Rate parameter for the underlying Poisson process, default is 1.15 for a mean frequeny of 2 at t=2.
size	Total sample size.
recruitment	Length of the recruitment period (in years), default is 3.
calendar	Calendar time of the end of the trial (in years), default is 5.
random.censor.rate	Rate parameter for independent random right censoring.
seed	Seed for reproducibility.

Value

A data frame in long format containing simulated composite endpoint data. Each subject may contributing multiple rows corresponding to recurrent events, a terminal event (death), or censoring. The data include:

- id: Subject identifier.
- e: Enrollment time on the calendar scale.
- event_time_cal: Cumulative event time on the calendar scale.
- status: Event indicator with values 2=recurrent event, 1=death, and 0=censoring.
- Z1, Z2: Simulated covariates used in the proportional mean model.
- tau_star: Subject-specific stopping time, the last event observed in [0, tau_star] is classified as death.
- death: Binary indicator for death.
- recurrent: Binary indicator for recurrent events.
- event: Binary event indicator, event = death + recurrent.
- calendar: Calendar time cutoff used to generate the returned data.
- lambda_0: Baseline Poisson process rate parameter.
- lambda_star: Rate parameter of an exponential distribution in generating tau_star.
- gamma_scale, gamma_shape: Parameters of the Gamma distribution used to generate subject-specific frailty terms.

References

Mao L, Lin DY. Semiparametric regression for the weighted composite endpoint of recurrent and terminal events. *Biostatistics*. 2016 Apr; **17(2)** :390-403.

Examples

```
# Generate one-sample composite endpoint data
df <- Onesample.generate.sequential(size = 200,
recruitment = 3, calendar = 5,
random.censor.rate = 0.05, seed = 1123)
```

`Solve.beta.given.power` *Function to solve for the treatment effect parameter to achieve target power by Monte Carlo simulation.*

Description

For each design scenario in `params`, this function solves for the treatment effect coefficient `beta.trt` that achieves the desired power using an iterative Monte Carlo calibration procedure. For scenarios labeled `TypeI`, the function sets `beta.trt = 0`. For scenarios labeled `Power`, it repeatedly simulates two-sample composites data, estimates calibration quantities (a power constant and variance) using either the generalized log-rank (`"LR"`) or generalized-t (`"GT"`) approach, updtaes `beta.trt` using `find.beta.trt()`, and iterates until convergence within `tol`.

Usage

```
Solve.beta.given.power(
  nsim = 1000,
  params,
  estimator,
  tol = 0.001,
  seed = NULL
)
```

Arguments

`nsim`	Integer giving the number of Monte Carlo replicates used for each iteration. Default is `1000`.
`params`	A data frame where each rows defines a simulation/design scenario. Must include a column `Type` with values `"TypeI"` or `"Power"`. For `"Power"` scenarios, the function expects the following columns: `placebo.lambda_0`, `grp.size`, `recruitment`, and `random.censor.rate`. Additional columns are carried through to the output.
`estimator`	Character string specifying which calibration method to use: `"LR"` for the generalized log-rank statistic or `"GT"` for the generalized-t statistic.
`tol`	Positive numeric value giving the convergence tolerance for the fixed-point iteration in `beta.trt`. The algorithm stops when `abs(beta.trt.old - beta.trt.new) < tol`. Default is `0.001`.
`seed`	Seed for reproducibility.

Value

A data.frame with the same rows as `params` and an additional column `beta.trt` containing the solved treatment effect coefficient. For `Type == "TypeI"`, `beta.trt` is set to `0`. For `Type == "Power"`, `beta.trt` is the converged solution from the Monte Carlo calibration procedure.

True.mu *Function to calculate the true value of the mean frequency function.*

Description

Computes the theoretical mean frequency function for the composite endpoint under the assumed data-generating mechanisim. The calculation uses the parameters stored in the simulated data.

Usage

```
True.mu(data, t)
```

Arguments

data	A data frame generated by `Onesample.generate.sequential()` (optionally after applying `Apply.calendar.censoring()`) containing composite endpoint data in long format. Each subject may contribute multiple rows corresponding to recurrent events, terminal event, or censoring.
t	A numeric vector specifying event times (time since enrollment) at which the true value of the mean frequency is evaluated.

Value

A numeric vector giving the true mean frequency evaluated at `t`.

References

Mao L, Lin DY. Semiparametric regression for the weighted composite endpoint of recurrent and terminal events. *Biostatistics*. 2016 Apr; **17(2)** :390-403.

Examples

```
df <- Onesample.generate.sequential(size = 200, recruitment = 3, calendar = 5,
random.censor.rate = 0.05, seed = 1123)
True.mu(data = df, t = c(1.5, 2.9, 4.6))
```

TwoSample.Boundary.TTFE *Function to calculate the sequential rejection boundaries for the time-to-first-event (TTFE) method.*

Description

Constructs two-sided group sequential critical boundaries for the two-sample TTFE log-rank Z-statistics at prespecified calendar times. The function converts stage-wise variance estimates into information fractions, allocates incremental Type I error using a spending function, and solves for the corresponding multivariate normal critical values under the canonical joint distribution assumption. It also returns stage-wise rejection indicators (with early stopping enforced).

Usage

```
TwoSample.Boundary.TTFE(result, spend, calendars, alpha, planned.n, Iunit)
```

Arguments

`result`	A list returned by `TwoSample.Z.Var.Estimator.Sequential.TTFE()`.
`spend`	A Type I error spending function. This should be a function of a single argument `t` (information fraction) returning cumulative alpha spent by information fraction `t` (e.g., `OBF`).
`calendars`	Numeric vector of interim analysis calendar times (in years).
`alpha`	Overall two-sided Type I error rate. Default typically `0.05`.
`planned.n`	Planned total sample size used to define the information scale.
`Iunit`	Information per subject at the final analysis.

Value

A list containing:

- `Z.stats`: Stage-wise TTFE Z-statistics.
- `vars`: Stage-wise variance estimates.
- `raw.information`: Stage-wise information fractions.
- `TTFE.bdry`: Two-sided critical boundaries at each analysis stage.
- `TTFE.reject`: Stage-wise rejection indicators (1 = reject at that stage, 0 otherwise), with later stages set to 0 after the first rejection.
- `nu`: Stage-wise information fractions after handling edge cases (e.g., reaching 100% information early).
- `pi`: Incremental Type I error allocated to each stage.
- `total.ns`: Total sample size contributing at each analysis stage (copied from `result`).

TwoSample.Constant.GT *Function to compute calibration constants for the two-sample generalized-t statistics.*

Description

Computes two quantities used for power/effect-size calibration of the two-sample generalized-t statistics. The function evaluates the generalized-t statistics up to a fixed event-time horizon `tau = 3`, then returns (i) an integral $\int_0^\tau K_{LR}(t)\, d\mu_1(t)$ based on the control group mean-frequency increment and (ii) the asymptotic variance of the integrated statistic.

Usage

```
TwoSample.Constant.GT(data)
```

Arguments

`data`	A data frame generated by `TwoSamaple.generate.sequential()`.

Value

- The calibration constant `power.const`. Defined as the integral of the generalized-t's weight function times the control group's mean frequency increment over `[0,3]`.
- The estimated asymptotic variance of the integrated two-sample generalized-t statistics over `[0,3]`.

TwoSample.Constant.LR *Function to compute calibration constants for the two-sample generalized log-rank statistics.*

Description

Computes two quantities used for power/effect-size calibration of the two-sample generalized log-rank statistics. The function evaluates the generalized log-rank statistics up to a fixed event-time horizon `tau = 3`, then returns (i) an integral $\int_0^\tau K_{LR}(t)\, d\mu_1(t)$ based on the control group mean-frequency increment and (ii) the asymptotic variance of the integrated statistic.

Usage

```
TwoSample.Constant.LR(data)
```

Arguments

`data`	A data frame generated by `TwoSamaple.generate.sequential()`.

Value

A numeric vector of length 2:

- The calibration constant `power.const`. Defined as the integral of the generalized log-rank statistics' weight function times the control group's mean frequency increment over `[0,3]`.
- The estimated asymptotic variance of the integrated two-sample generalized log-rank statistics over `[0,3]`.

TwoSample.Estimator.GT.sequential

Function to calculate the two-sample generalized-t statistic for composite endpoint under sequential monitoring.

Description

Computes a two-sample generalized t-test statistic for composite endpoints consisting of recurrent events and a terminal event, using data observed up to a given calendar time. Event times are converted from the calendar-time scale to the event-time scale (time since enrollment).

Usage

```
TwoSample.Estimator.GT.sequential(data, tau = 3)
```

Arguments

`data`	A data frame generated by `TwoSample.generate.sequential()` (optionally after applying `Apply.calendar.censoring.2()`) containing simulated two-sample composite endpoint data.
`tau`	Positive numeric value specifying the upper bound of event time for the integration. Default is `3`.

Value

A list with components:

- `Q`: Value of the generalized t-test statistics integrated over `[0, tau]`.
- `var`: Estimated asymptotic variance of `Q`.
- `const`: Scaling constant used in the variance estimation.

Examples

```
# Two-sample generalized-t statistic: null hypothesis
df <- TwoSample.generate.sequential(sizevec = c(200, 200),
beta.trt = 0, calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
TwoSample.Estimator.GT.sequential(data = df, tau = 3)
# Two-sample generalized-t statistic: alternative hypothesis
```

```
df2 <- TwoSample.generate.sequential(sizevec = c(200, 200),
beta.trt = 0.8, calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
TwoSample.Estimator.GT.sequential(data = df2, tau = 3)
```

`TwoSample.Estimator.LR.sequential`

Function to calculate the two-sample generalized log-rank statistic for composite endpoint under sequential monitoring.

Description

Computes a two-sample generalized log-rank test statistic for composite endpoints consisting of recurrent events and a terminal event, using data observed up to a given calendar time. Event times are converted from the calendar-time scale to the event-time scale (time since enrollment). The statistics integrates the difference between estimated mean frequency increments across groups with a risk-set based weight function.

Usage

```
TwoSample.Estimator.LR.sequential(data, tau = 3)
```

Arguments

`data`	A data frame generated by `TwoSample.generate.sequential()` (optionally after applying `Apply.calendar.censoring.2()`) containing simulated two-sample composite endpoint data.
`tau`	Positive numeric value specifying the upper bound of event time for the integration. Default is `3`.

Value

A list with components:

- `Q`: Value of the generalized log-rank statistics integrated over `[0, tau]`.
- `var`: Estimated asymptotic variance of `Q`.
- `const`: Scaling constant used in the variance estimation.

Examples

```
# Two-sample generalized log-rank test: null hypothesis
df <- TwoSample.generate.sequential(sizevec = c(200, 200),
beta.trt = 0, calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
TwoSample.Estimator.LR.sequential(data = df, tau = 3)
# Two-sample generalized log-rank test: alternative hypothesis
df2 <- TwoSample.generate.sequential(sizevec = c(200, 200),
```

```
beta.trt = 0.8, calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
TwoSample.Estimator.LR.sequential(data = df2, tau = 3)
```

`TwoSample.Estimator.TTFE.sequential`

Function to calculate the Z statistics and variance for the time-to-first-event (TTFE) method.

Description

Computes a two-sample TTFE test statistics by retaining only the first observed event for each subject and applying a standard log-rank test. This function is intended as a comparator or benchmark to the mean-frequency based estimator implemented elsewhere in the package.

Usage

```
TwoSample.Estimator.TTFE.sequential(data)
```

Arguments

`data`	A data frame generated by `TwoSamaple.generate.sequential()`.

Value

A list with:

- `Z.stat`: The ordinary log-rank statistics on the Z-scale, defined as the square root of the chi-square statistic from `survdiff()`.
- `var`: The estimated variance of the test statistic. extracted from the log-rank test variance matrix.

`TwoSample.generate.sequential`

Function to simulate two-sample composite endpoint data under staggered entry.

Description

Simulate two-sample composite endpoints data with recurrent events and a terminal event under two time scales: event time `t` and calendar time `s`. A uniform recruitment period is assumed, and the function returns all observed data available at a specified calendar time. Recurrent event occurrences are generated from an underlying Poisson process with subject-specific Gamma frailty.

Usage

```
TwoSample.generate.sequential(
  lambda_0vec = c(1.15, 1.15),
  sizevec,
  beta.trt,
  calendar = 5,
  recruitment = 3,
  random.censor.rate,
  seed
)
```

Arguments

`lambda_0vec`	Numeric vector of length 2 giving the baseline recurrent event rate parameters for two arms, default is `c(1.15, 1.15)`.
`sizevec`	Integer vector of size 2 giving the group sizes.
`beta.trt`	Numeric value giving the treatment effect coefficient applied to the treatment indicator (`Z1`) in the proportional mean model (arm 2 vs arm 1).
`calendar`	Calendar time of the end of the trial (in years), default is `5`.
`recruitment`	Length of the recruitment period (in years), default is `3`.
`random.censor.rate`	Rate parameter for independent random right censoring.
`seed`	Seed for reproducibility.

Value

A data frame in long format containing simulated composite endpoint data. Each subject may contributing multiple rows corresponding to recurrent events, a terminal event (death), or censoring. The data include:

- `group`: Arm indicator (`1` = control, `2`= treatment).
- `id`: Subject identifier (unique across both arms).
- `e`: Enrollment time on the calendar scale.
- `event_time_cal`: Cumulative event time on the calendar scale.
- `status`: Event indicator with values `2`=recurrent event, `1`=death, and `0`=censoring.
- `Z1`, `Z2`: Simulated covariates, where `Z1` is the treatment indicator (`0` for arm 1, `1` for arm 2).
- `tau_star`: Subject-specific stopping time, the last event observed in `[0, tau_star]` is classified as death.
- `death`: Binary indicator for death.
- `recurrent`: Binary indicator for recurrent events.
- `event`: Binary event indicator, `event = death + recurrent`.
- `calendar`: Calendar time cutoff used to generate the returned data.
- `lambda_0`: Baseline Poisson process rate parameter.
- `lambda_star`: Rate parameter of an exponential distribution in generating `tau_star`.
- `gamma_scale`, `gamma_shape`: Parameters of the Gamma distribution used to generate subject-specific frailty terms.

References

Mao L, Lin DY. Semiparametric regression for the weighted composite endpoint of recurrent and terminal events. *Biostatistics*. 2016 Apr; **17(2)** :390-403.

Examples

```
# Generate two-sample composite endpoint data
df <- TwoSample.generate.sequential(sizevec = c(200, 200),
beta.trt = 0.8, calendar = 5, recruitment = 3,
random.censor.rate = 0.05, seed = 2026)
```

TwoSample.Q.Cov.Estimator.Sequential.GT

Function to calculate stage-wise test statistics, variances, and correlation for two-sample generalized-t statistics.

Description

Computes the stage-wise generalized-t statistics and their variance estimates at a set of interim analysis calendar times. At each analysis, administrative censoring is applied at the specified calendar time, event times are converted from the calendar-time scale to the event-time scale (time since enrollment), and the generalized-t statistics is evaluated over `[0, tau]`. When multiple analysis are requested, the function also estimates the correlation matrix of the stage-wise statistics.

Usage

```
TwoSample.Q.Cov.Estimator.Sequential.GT(data, tau = 3, calendars)
```

Arguments

`data`	A data.frame generated by `TwoSample.generate.sequential()`.
`tau`	Positive numeric value specifying the upper bound of event time (time since enrollment) for integration of the statistic. Default is `3`.
`calendars`	Numeric vector of interim analysis calendar times (in years) at which to compute stage-wise statistics and variance estimates.

Value

A list containing stage-wise estimates. If `length(calendars) > 1`, the returned list includes:

- `Qs`: Numeric vector of stage-wise generalized-t statistics evaluated at each calendar time in `calendars`.
- `vars`: Numeric vector of estimated variances corresponding to `Qs`.
- `total.ns`: Numeric vector giving the total enrolled sample size contributing data at each calendar time.

- corr.matrix: Estimated correlation matrix of the stage-wise statistics.
- nss: List of length length(calendars) giving the group-specific sample sizes at each analysis.

If length(calendars) == 1, the list contains Qs, vars, and total.ns.

Examples

```
df <- TwoSample.generate.sequential(sizevec = c(200, 200), beta.trt = 0,
calendar = 5, recruitment = 3, random.censor.rate = 0.05, seed = 2026)
TwoSample.Q.Cov.Estimator.Sequential.GT(data = df, calendars = c(2.5, 3.5, 4.5))
```

TwoSample.Q.Cov.Estimator.Sequential.LR

Function to calculate stage-wise test statistics, variances, and correlation for two-sample generalized log-rank statistics.

Description

Computes the stage-wise generalized log-rank statistics and their variance estimates at a set of interim analysis calendar times. At each analysis, administrative censoring is applied at the specified calendar time, event times are converted from the calendar-time scale to the event-time scale (time since enrollment), and the generalized log-rank statistics is evaluated over [0, tau]. When multiple analysis are requested, the function also estimates the correlation matrix of the stage-wise statistics.

Usage

```
TwoSample.Q.Cov.Estimator.Sequential.LR(data, tau = 3, calendars)
```

Arguments

data	A data.frame generated by TwoSample.generate.sequential().
tau	Positive numeric value specifying the upper bound of event time (time since enrollment) for integration of the statistic. Default is 3.
calendars	Numeric vector of interim analysis calendar times (in years) at which to compute stage-wise statistics and variance estimates.

Value

A list containing stage-wise estimates. If length(calendars) > 1, the returned list includes:

- Qs: Numeric vector of stage-wise generalized log-rank statistics evaluated at each calendar time in calendars.
- vars: Numeric vector of estimated variances corresponding to Qs.
- total.ns: Numeric vector giving the total enrolled sample size contributing data at each calendar time.

- corr.matrix: Estimated correlation matrix of the stage-wise statistics.
- nss: List of length length(calendars) giving the group-specific sample sizes at each analysis.

If length(calendars) == 1, the list contains Qs, vars, and total.ns.

Examples

```
df <- TwoSample.generate.sequential(sizevec = c(200, 200), beta.trt = 0,
calendar = 5, recruitment = 3, random.censor.rate = 0.05, seed = 2026)
TwoSample.Q.Cov.Estimator.Sequential.LR(data = df, calendars = c(2.5, 3.5, 4.5))
```

TwoSample.Wald.and.Boundary

Function to calculate the Wald statistics and group sequential boundaries for two-sample monitoring.

Description

Computes stage-wise Wald statistics from estimated two-sample statistics and their variances, determines information fractions, allocates Type I error using a user specified spending function, and constructs corresponding two-sided rejection boundaries. The function supports both a consistent correlation approach (supplied in Q.cov.est) and a canonical correlation approach based on information fractions.

Usage

```
TwoSample.Wald.and.Boundary(
  Q.cov.est,
  spend,
  calendars,
  alpha,
  planned.n,
  Iunit
)
```

Arguments

Q.cov.est	A list containing stage-wise estimates and covariances, either returned by TwoSample.Estimator.LR.sec or TwoSample.Estimator.GT.sequential().
spend	A function specifying the cumulative Type I error spending function $\alpha(t)$ evaluated at information fraction t. For example, OBF.
calendars	Numeric vector of analysis calendar times (in years), defining the planned monitoring schedule and number of analysis.
alpha	Overall two-sided Type I error.
planned.n	Planned total sample size at the final analysis.
Iunit	Information per subject (or per unit of sample size) used to scale the information fractions.

Value

A list with components:

- `Qs`: Stage-wise test statistics.
- `vars`: Stage-wise variance estimates for `Qs`.
- `raw.information`: Information fractions prior to any adjustments for early completion or skipped analysis.
- `Wald`: Stage-wise Wald statistics `Qs/sqrt(vars)`.
- `consistent.bdry`: Two-sided rejection boundaries computed using the consistent correlation matrix.
- `canonical.bdry`: Two-sided rejection boundaries computed using the canonical correlation matrix.
- `consistent.reject`: Indicator vector for boundary crossing under the consistent approach (only the the first crossing is retained).
- `canonical.reject`: Indicator vector for boundary crossing under the canonical approach (only the the first crossing is retained).
- `nu`: Final information fractions used for boundary construction.
- `pi`: Incremental Type I error allocated to each analysis.
- `total.ns`: Total accrued sample size at each analysis.

TwoSample.Z.Var.Estimator.Sequential.TTFE

Function to calculate the two-sample time-to-first-event (TTFE) statistics and variances across multiple calendar times.

Description

Computes sequential Z-statistics and their corresponding variance estimates for a two-sample time-to-first-event (TTFE) analysis at a set of prespecified calendar analysis times. At each calendar time, administrative censoring is applied, event times are converted from the calendar scale to the event-time scale, and a standard log-rank test is performed using only the first observed event per subject.

Usage

```
TwoSample.Z.Var.Estimator.Sequential.TTFE(data, tau = NULL, calendars)
```

Arguments

`data`	A data frame containing two-sample composite endpoint data, typically generated by `TwoSample.generate.sequential()`.
`tau`	Optional upper bound for the event-time horizon. This argument is currently not used and is included for interface consistency with other sequential estimators in the package.
`calendars`	A numeric vector of calendar times at which interim analyses are conducted. Each value is treated as an administrative censoring time.

Value

A list with components:

- `Z.stats`: A numeric vector of log-rank Z-statistics evaluated at each calendar analysis time.
- `vars`: A numeric vector of estimated variances of the Z-statistics at each analysis time.
- `total.ns`: A numeric vector giving the total number of subjects contributing data at each calendar analysis time.

Index