Package 'gsDesignNB'

February 16, 2026

Version 0.2.6

Title Sample Size and Simulation for Negative Binomial Outcomes

Description Provides tools for planning and simulating recurrent event trials with overdispersed count endpoints analyzed using negative binomial (or Poisson) rate models. Implements sample size and power calculations for fixed designs with variable accrual, dropout, maximum follow-up, and event gaps, including methods of Zhu and Lakkis (2014) <doi:10.1002/sim.5947> and Friede and Schmidli (2010) <doi:10.3414/ME09-02-0060>. Supports group sequential designs by adding calendar-time analysis schedules compatible with the 'gsDesign' package and by estimating blinded information at interim looks. Includes simulation utilities for recurrent events (including seasonal rates), interim data truncation, and Wald-based inference for treatment rate ratios.

License GPL (>= 3)

URL https://keaven.github.io/gsDesignNB/, https://github.com/keaven/gsDesignNB

BugReports https://github.com/keaven/gsDesignNB/issues

Encoding UTF-8

Depends R (>= 3.5.0)

Imports data.table, gsDesign, simtrial, stats, MASS

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, dplyr, gt, scales

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Keaven Anderson [aut, cre],
Nan Xiao [ctb],
Hongtao Zhang [ctb],
Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph] (ROR: <https://ror.org/02891sr49>)

Maintainer Keaven Anderson <keaven_anderson@merck.com>

Repository CRAN

Date/Publication 2026-02-16 17:40:09 UTC

Contents

`blinded_ssr` *Blinded sample size re-estimation for recurrent events*

Description

Estimates the blinded dispersion and event rate from aggregated interim data and calculates the required sample size to maintain power, assuming the planned treatment effect holds. This function supports constant rates (Friede & Schmidli 2010) and accommodates future extensions for time-varying rates (Schneider et al. 2013) by using the exposure-adjusted rate.

Usage

```
blinded_ssr(
  data,
  ratio = 1,
  lambda1_planning,
  lambda2_planning,
  rr0 = 1,
  power = 0.8,
  alpha = 0.025,
  method = "friede",
  accrual_rate,
  accrual_duration,
  trial_duration,
  dropout_rate = 0,
  max_followup = NULL,
  event_gap = NULL
)
```

Arguments

`data`	A data frame containing the blinded interim data. Must include columns `events` (number of events) and `tte` (total exposure/follow-up time). This is typically the output of `cut_data_by_date()`.
`ratio`	Planned allocation ratio (experimental / control). Default is 1.
`lambda1_planning`	Planned event rate for the control group used in original calculation.
`lambda2_planning`	Planned event rate for the experimental group used in original calculation.
`rr0`	Rate ratio under the null hypothesis (lambda2/lambda1). Default is 1.
`power`	Target power (1 - beta). Default is 0.8.
`alpha`	One-sided significance level. Default is 0.025.
`method`	Method for sample size recalculation. Currently "friede" (Friede & Schmidli 2010) is implemented, which uses the blinded nuisance parameter estimates.
`accrual_rate`	Vector of accrual rates (patients per unit time).
`accrual_duration`	Vector of durations for each accrual rate. Must be same length as `accrual_rate`.
`trial_duration`	Total planned duration of the trial.
`dropout_rate`	Dropout rate (hazard rate). Default is 0.
`max_followup`	Maximum follow-up time for any patient. Default is NULL (infinite).
`event_gap`	Gap duration after each event during which no new events are counted. Default is NULL (no gap).

Value

A list containing:

n_total_blinded Re-estimated total sample size using blinded estimates.

dispersion_blinded Estimated dispersion parameter (k) from blinded data.

lambda_blinded Estimated overall event rate from blinded data.

info_fraction Estimated information fraction at interim (blinded information / target information).

blinded_info Estimated statistical information from the blinded interim data.

target_info Target statistical information required for the planned power.

References

Friede, T., & Schmidli, H. (2010). Blinded sample size reestimation with count data: methods and applications in multiple sclerosis. *Statistics in Medicine*, 29(10), 1145–1156. doi:10.1002/sim.3861

Schneider, S., Schmidli, H., & Friede, T. (2013). Blinded sample size re-estimation for recurrent event data with time trends. *Statistics in Medicine*, 32(30), 5448–5457. doi:10.1002/sim.5977

Examples

```
interim <- data.frame(events = c(1, 2, 1, 3), tte = c(0.8, 1.0, 1.2, 0.9))
blinded_ssr(
  interim,
  ratio = 1,
  lambda1_planning = 0.5,
  lambda2_planning = 0.3,
  power = 0.8,
  alpha = 0.025,
  accrual_rate = 10,
  accrual_duration = 12,
  trial_duration = 18
)
```

---

calculate_blinded_info

Calculate blinded statistical information

Description

Estimates the blinded dispersion and event rate from aggregated interim data and calculates the observed statistical information for the log rate ratio, assuming the planned allocation ratio and treatment effect.

Usage

```
calculate_blinded_info(
  data,
  ratio = 1,
  lambda1_planning,
  lambda2_planning,
  event_gap = NULL
)
```

Arguments

`data`	A data frame containing the blinded interim data. Must include columns `events` (number of events) and `tte` (total exposure=follow-up time).
`ratio`	Planned allocation ratio (experimental / control). Default is 1.
`lambda1_planning`	Planned event rate for the control group.
`lambda2_planning`	Planned event rate for the experimental group.
`event_gap`	Optional. Gap duration (numeric) to adjust planning rates if provided. If provided, planning rates are adjusted as lambda / (1 + lambda * gap).

Value

A list containing:

blinded_info Estimated statistical information.

dispersion_blinded Estimated dispersion parameter (k).

lambda_blinded Estimated overall event rate.

lambda1_adjusted Re-estimated control rate.

lambda2_adjusted Re-estimated experimental rate.

Examples

```
interim <- data.frame(events = c(1, 2, 1, 3), tte = c(0.8, 1.0, 1.2, 0.9))
calculate_blinded_info(
  interim,
  ratio = 1,
  lambda1_planning = 0.5,
  lambda2_planning = 0.3
)
```

check_gs_bound *Check group sequential bounds*

Description

Updates the group sequential design boundaries based on observed information and checks if boundaries have been crossed.

Usage

```
check_gs_bound(sim_results, design, info_scale = c("blinded", "unblinded"))
```

Arguments

sim_results	Data frame of simulation results (from sim_gs_nbinom()).
design	The planning gsNB object.
info_scale	Character. "blinded" (default) or "unblinded" information to use for bounds.

Value

A data frame with added columns:

cross_upper Logical, true if upper bound crossed (efficacy)

cross_lower Logical, true if lower bound crossed (futility)

Examples

```
design <- gsDesign::gsDesign(k = 2, n.fix = 100, test.type = 2, timing = c(0.5, 1))
sim_df <- data.frame(
  sim = c(1, 1, 2, 2),
  analysis = c(1, 2, 1, 2),
  z_stat = c(2.5, NA, -0.2, 2.2),
  blinded_info = c(50, 100, 50, 100),
  unblinded_info = c(50, 100, 50, 100)
)
check_gs_bound(sim_df, design)
```

`compute_info_at_time` *Compute statistical information at analysis time*

Description

Computes the statistical information for the log rate ratio at a given analysis time, accounting for staggered enrollment and varying exposure times.

Usage

```
compute_info_at_time(
  analysis_time,
  accrual_rate,
  accrual_duration,
  lambda1,
  lambda2,
  dispersion,
  ratio = 1,
  dropout_rate = 0,
  event_gap = 0,
  max_followup = Inf
)
```

Arguments

`analysis_time`	The calendar time of the analysis.
`accrual_rate`	The enrollment rate (subjects per time unit).
`accrual_duration`	The duration of the enrollment period.
`lambda1`	Event rate for group 1 (control).
`lambda2`	Event rate for group 2 (treatment).
`dispersion`	The negative binomial dispersion parameter.
`ratio`	Allocation ratio (n2/n1). Default is 1.
`dropout_rate`	Dropout rate (hazard rate). Default is 0.
`event_gap`	Gap duration after each event during which no new events are counted. Default is 0.
`max_followup`	Maximum follow-up time per subject. Exposure time is truncated at this value. Default is `Inf` (no truncation).

Value

The statistical information (inverse of variance) at the analysis time.

Examples

```
compute_info_at_time(
  analysis_time = 12,
  accrual_rate = 10,
  accrual_duration = 10,
  lambda1 = 0.5,
  lambda2 = 0.3,
  dispersion = 0.1
)
```

`cut_completers` *Cut data for completers analysis*

Description

Subsets the data to all subjects randomized by the specified date, and prepares the data for analysis. This is a wrapper for `cut_data_by_date()` typically used with a date determined by `cut_date_for_completers()`.

Usage

```
cut_completers(data, cut_date, event_gap = 0)
```

Arguments

`data`	Data generated by `nb_sim()`.
`cut_date`	Calendar time (relative to trial start) at which to cut the data.
`event_gap`	Gap duration after each event during which no new events are counted. Can be a numeric value (default `0`) or a function returning a numeric value. The time at risk is reduced by the sum of these gaps (truncated by the cut date).

Value

A data frame with one row per subject randomized prior to `cut_date`. Contains the truncated follow-up time (`tte`) and total number of observed events (`events`).

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 20)
# Find date when 5 subjects have completed
date_5 <- cut_date_for_completers(sim, 5)
# Get analysis dataset for this cut date (includes partial follow-up)
cut_completers(sim, date_5)
```

`cut_data_by_date` *Cut simulated trial data at a calendar date*

Description

Censors follow-up at a specified calendar time and aggregates events per subject. Returns one row per subject randomized before the cut date, with the total number of observed events and follow-up times.

Usage

```
cut_data_by_date(data, cut_date, event_gap = 0, ...)

## Default S3 method:
cut_data_by_date(data, cut_date, event_gap = 0, ...)

## S3 method for class 'nb_sim_data'
cut_data_by_date(data, cut_date, event_gap = 0, ...)

## S3 method for class 'nb_sim_seasonal'
cut_data_by_date(data, cut_date, event_gap = 0, ...)
```

Arguments

`data`	Data generated by `nb_sim()`.
`cut_date`	Calendar time (relative to trial start) at which to censor follow-up.
`event_gap`	Gap duration after each event during which no new events are counted. Can be a numeric value (default `0`) or a function returning a numeric value. The time at risk is reduced by the sum of these gaps (truncated by the cut date).
`...`	Additional arguments passed to methods.

Value

A data frame with one row per subject randomized prior to `cut_date` containing:

id Subject identifier

treatment Treatment group

enroll_time Time of enrollment relative to trial start

tte Time at risk (total follow-up minus event gap periods)

tte_total Total follow-up time (calendar time, not adjusted for gaps)

events Number of observed events

A data frame with one row per subject randomized prior to `cut_date`. This method stops with an error for unsupported classes.

A data frame with one row per subject randomized prior to `cut_date`. Includes total events and follow-up time within the cut window.

A data frame with one row per subject randomized prior to `cut_date`. Includes season and follow-up time within the cut window.

Methods (by class)

- `cut_data_by_date(default)`: Default method.
- `cut_data_by_date(nb_sim_data)`: Method for `nb_sim` data.
- `cut_data_by_date(nb_sim_seasonal)`: Method for `nb_sim_seasonal` data.

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 20)
cut_data_by_date(sim, cut_date = 1)
```

`cut_date_for_completers` *Find calendar date for target completer count*

Description

Finds the calendar time (since start of randomization) at which a specified number of subjects have completed their follow-up.

Usage

```
cut_date_for_completers(data, target_completers)
```

Arguments

`data`	A data frame of simulated data, typically from `nb_sim()` or `nb_sim_seasonal()`.
`target_completers`	Integer. The target number of completers.

Value

Numeric. The calendar date when `target_completers` is achieved. If the dataset contains fewer than `target_completers` completers, returns the maximum calendar time in the dataset and prints a message.

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 20)
cut_date_for_completers(sim, target_completers = 5)
```

---

estimate_nb_mom *Method of Moments Estimation for Negative Binomial Parameters*

Description

Estimates the event rate(s) and common dispersion parameter (k) for negative binomial count data using the method of moments. This is a robust alternative to Maximum Likelihood Estimation (MLE), especially when MLE fails to converge or produces boundary estimates.

Usage

```
estimate_nb_mom(data, group = NULL)
```

Arguments

`data`	A data frame containing the data. Must include columns `events` (number of events) and `tte` (total exposure/follow-up time).
`group`	Optional character string specifying the grouping column name (e.g., "treatment"). If provided, rates are estimated separately for each group, while a common dispersion parameter is estimated across groups. If NULL (default), a single rate and dispersion are estimated (blinded case).

Details

The method of moments estimator for the dispersion parameter k is derived by equating the theoretical variance to the observed second central moment, accounting for varying exposure times.

For a given group with rate λ, the expected count for subject i is $\mu_i = \lambda t_i$. The variance is $V_i = \mu_i + k\mu_i^2$. The estimator is calculated as:

$$\hat{k} = \max\left(0, \frac{\sum(y_i - \hat{\mu}_i)^2 - \sum y_i}{\sum \hat{\mu}_i^2}\right)$$

where y_i is the number of events, t_i is the exposure time, and $\hat{\mu}_i = \hat{\lambda} t_i$ is the estimated expected count.

When multiple groups are present, the numerator and denominator are summed across all groups to estimate a common k.

Value

A list containing:

`lambda`	Estimated event rate(s). A single numeric value if `group` is NULL, or a named vector if `group` is provided.
`dispersion`	Estimated common dispersion parameter (k).

Examples

```
# Blinded estimation (single group)
df <- data.frame(events = c(1, 2, 0, 3), tte = c(1, 1.2, 0.5, 1.5))
estimate_nb_mom(df)

# Unblinded estimation (two groups)
df_group <- df
df_group$group <- c("A", "A", "B", "B")
estimate_nb_mom(df_group, group = "group")
```

`get_analysis_date` *Find calendar date for target event count*

Description

Finds the calendar time (since start of randomization) at which a specified total number of events is reached in the simulated dataset.

Usage

```
get_analysis_date(data, planned_events, event_gap = 5/365.25)
```

Arguments

`data`	A data frame of simulated data, typically from `nb_sim()`.
`planned_events`	Integer. The target number of events.
`event_gap`	Gap duration after each event during which no new events are counted. Can be a numeric value (default `5 / 365.25`) or a function returning a numeric value.

Value

Numeric. The calendar date when `planned_events` is achieved. If the dataset contains fewer than `planned_events`, returns the maximum calendar time in the dataset and prints a message.

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 40)
get_analysis_date(sim, planned_events = 15)
```

---

get_cut_date *Determine analysis date based on criteria*

Description

Finds the earliest calendar date at which all specified criteria are met. Criteria can include a specific calendar date, a target number of events, a target number of completers, or a target amount of blinded information.

Usage

```
get_cut_date(
  data,
  planned_calendar = NULL,
  target_events = NULL,
  target_completers = NULL,
  target_info = NULL,
  event_gap = 0,
  ratio = 1,
  lambda1 = NULL,
  lambda2 = NULL,
  min_date = 0,
  max_date = Inf
)
```

Arguments

`data`	A data frame of simulated data (from `nb_sim()`).
`planned_calendar`	Numeric. Target calendar time.
`target_events`	Integer. Target number of observed events.
`target_completers`	Integer. Target number of subjects with complete follow-up.
`target_info`	Numeric. Target blinded information.
`event_gap`	Numeric. Gap duration for event counting and info calculation.
`ratio`	Numeric. Randomization ratio (experimental/control) for info calculation.

lambda1	Numeric. Planned control rate for info calculation.
lambda2	Numeric. Planned experimental rate for info calculation.
min_date	Numeric. Minimum possible date (e.g., 0 or previous analysis time).
max_date	Numeric. Maximum possible date (e.g., trial duration).

Value

Numeric. The calendar date satisfying the criteria. If criteria cannot be met within max_date (or data limits), returns max_date (or max data time).

Examples

```
set.seed(456)
enroll_rate <- data.frame(rate = 15, duration = 1)
fail_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.6, 0.4)
)
sim_data <- nb_sim(enroll_rate, fail_rate, max_followup = 1, n = 20)
get_cut_date(sim_data, planned_calendar = 0.5, target_events = 5, event_gap = 0)
```

---

gsNBCalendar *Group sequential design for negative binomial outcomes*

Description

Creates a group sequential design for negative binomial outcomes based on sample size calculations from sample_size_nbinom().

Usage

```
gsNBCalendar(
  x,
  k = 3,
  test.type = 4,
  alpha = 0.025,
  beta = 0.1,
  astar = 0,
  delta = 0,
  sfu = gsDesign::sfHSD,
  sfupar = -4,
  sfl = gsDesign::sfHSD,
  sflpar = -2,
  tol = 1e-06,
  r = 18,
  usTime = NULL,
  lsTime = NULL,
  analysis_times = NULL
)
```

Arguments

x	An object of class sample_size_nbinom_result from sample_size_nbinom().
k	Number of analyses (interim + final). Default is 3.
test.type	Test type as in gsDesign::gsDesign(): **1** One-sided **2** Two-sided symmetric **3** Two-sided, asymmetric, binding futility bound, beta-spending **4** Two-sided, asymmetric, non-binding futility bound, beta-spending **5** Two-sided, asymmetric, binding futility bound, lower spending **6** Two-sided, asymmetric, non-binding futility bound, lower spending Default is 4.
alpha	Type I error (one-sided). Default is 0.025.
beta	Type II error (1 - power). Default is 0.1.
astar	Allocated Type I error for lower bound for test.type = 5 or 6. Default is 0.
delta	Standardized effect size. Default is 0 (computed from design).
sfu	Spending function for upper bound. Default is gsDesign::sfHSD.
sfupar	Parameter for upper spending function. Default is -4.
sfl	Spending function for lower bound. Default is gsDesign::sfHSD.
sflpar	Parameter for lower spending function. Default is -2.
tol	Tolerance for convergence. Default is 1e-06.
r	Integer controlling grid size for numerical integration. Default is 18.
usTime	Spending time for upper bound (optional).
lsTime	Spending time for lower bound (optional).
analysis_times	Vector of calendar times for each analysis. Must have length k. These times are stored in the T element and displayed by gsDesign::gsBoundSummary().

Value

An object of class gsNB which inherits from gsDesign and sample_size_nbinom_result. While the final sample size would be planned total enrollment, interim analysis sample sizes are the expected number enrolled at the times specified in analysis_times. Output value contains all elements from gsDesign::gsDesign() plus:

nb_design The original sample_size_nbinom_result object

n1 A vector with sample size per analysis for group 1

n2 A vector with sample size per analysis for group 2

n_total A vector with total sample size per analysis

events A vector with expected total events per analysis

events1 A vector with expected events per analysis for group 1

events2 A vector with expected events per analysis for group 2

exposure A vector with expected average calendar exposure per analysis

exposure_at_risk1 A vector with expected at-risk exposure per analysis for group 1

exposure_at_risk2 A vector with expected at-risk exposure per analysis for group 2

variance A vector with variance of log rate ratio per analysis

T Calendar time at each analysis (if `analysis_times` provided)

Note that `n.I` in the returned object represents the statistical information at each analysis.

References

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

Examples

```
# First create a sample size calculation
nb_ss <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.9,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)

# Then create a group sequential design with analysis times
gs_design <- gsNBCalendar(nb_ss,
  k = 3, test.type = 4,
  analysis_times = c(10, 18, 24)
)
```

---

mutze_test *Wald test for treatment effect using negative binomial model (Mutze et al.)*

Description

Fits a negative binomial (or Poisson) log-rate model to the aggregated subject-level data produced by `cut_data_by_date()`. The method matches the Wald test described by Mutze et al. (2019) for comparing treatment arms with recurrent event outcomes.

Usage

```
mutze_test(
  data,
  method = c("nb", "poisson"),
  conf_level = 0.95,
  sided = 1,
  poisson_threshold = 1000
)

## S3 method for class 'mutze_test'
print(x, ...)
```

Arguments

data	A data frame with at least the columns treatment, events, and tte (follow-up time). Typically output from cut_data_by_date().
method	Type of model to fit: "nb" (default) uses a negative binomial GLM via MASS::glm.nb(), "poisson" fits a Poisson GLM.
conf_level	Confidence level for the rate ratio interval. Default 0.95.
sided	Number of sides for the test: 1 (default) or 2.
poisson_threshold	When method = "nb", the model falls back to Poisson regression if theta (the NB shape parameter) is outside the range [1/poisson_threshold, poisson_threshold]. Very large theta indicates near-Poisson data, while very small theta indicates extreme overdispersion with unstable estimates. Default is 1000.
x	An object of class mutze_test.
...	Additional arguments (currently ignored).

Value

An object of class mutze_test containing the fitted model summary with elements:

- method: A string indicating the test method used.
- estimate: log rate ratio (experimental vs control).
- se: standard error for the log rate ratio.
- z: Wald statistic.
- p_value: one-sided or two-sided p-value.
- rate_ratio: estimated rate ratio and its confidence interval.
- dispersion: estimated dispersion (theta) when method = "nb".
- group_summary: observed subjects/events/exposure per treatment.

Invisibly returns the input object.

Methods (by generic)

- print(mutze_test): Print method for mutze_test objects.

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 40)
cut <- cut_data_by_date(sim, cut_date = 1.5)
mutze_test(cut)
```

nb_sim *Simulate recurrent events with fixed follow-up*

Description

Simulates recurrent events for a clinical trial with piecewise constant enrollment, exponential failure rates (Poisson process), and piecewise exponential dropout.

Usage

```
nb_sim(
  enroll_rate,
  fail_rate,
  dropout_rate = NULL,
  max_followup = NULL,
  n = NULL,
  block = c(rep("Control", 2), rep("Experimental", 2)),
  event_gap = 0
)
```

Arguments

enroll_rate	A data frame with columns rate and duration defining the piecewise constant enrollment rates.
fail_rate	A data frame with columns treatment and rate defining the exponential failure rate for each treatment group. Optionally, a dispersion column can be provided to generate data from a negative binomial distribution. The dispersion parameter k is such that $\mathrm{Var}(Y) = \mu + k\mu^2$.
dropout_rate	A data frame with columns treatment, rate, and duration defining the piecewise constant dropout rates.
max_followup	Numeric. Maximum duration of follow-up for each individual (relative to their randomization time).
n	Total sample size. If NULL, it is estimated from enroll_rate. If provided, enrollment stops when n subjects are recruited.
block	Block vector for treatment allocation. Default is c(rep("Control", 2), rep("Experimental", 2)). If NULL, simple randomization is used (treatments are assigned with equal probability). If provided, it specifies the block structure, for example, c(rep("A", 2), rep("B", 2)) assigns 2 to group A and 2 to group B in each block.
event_gap	Numeric. Gap duration after each event during which no new events are counted. Default is 0.

Details

The simulation generates data consistent with the negative binomial models described by Friede and Schmidli (2010) and Mütze et al. (2019). Specifically, it simulates a Gamma-distributed frailty variable for each individual (if dispersion > 0), which acts as a multiplier for that individual's event rate. Events are then generated according to a Poisson process with this subject-specific rate.

More explicitly, for a subject with baseline rate λ and exposure time t, the model used here is a Gamma–Poisson mixture:

$$\Lambda_i \sim \text{Gamma}(\text{shape} = 1/k,\ \text{scale} = k\lambda), \quad Y_i \mid \Lambda_i \sim \text{Poisson}(\Lambda_i t).$$

Marginally, Y_i follows a negative binomial distribution with $\text{E}[Y_i] = \mu = \lambda t$ and $\text{Var}(Y_i) = \mu + k\mu^2$. This k is the package dispersion parameter (and corresponds to $1/\theta$ in `MASS::glm.nb()` terminology).

Value

A data frame (tibble) with columns:

id Subject identifier

treatment Treatment group

enroll_time Time of enrollment relative to trial start

tte Time to event or censoring relative to randomization

calendar_time Calendar time of event or censoring (enroll_time + tte)

event Binary indicator: 1 for event, 0 for censoring

Multiple rows per subject are returned (one for each event, plus one for the final censoring time).

References

Friede, T., & Schmidli, H. (2010). Blinded sample size reestimation with count data: methods and applications in multiple sclerosis. *Statistics in Medicine*, 29(10), 1145–1156. doi:10.1002/sim.3861

Mütze, T., Glimm, E., Schmidli, H., & Friede, T. (2019). Group sequential designs for negative binomial outcomes. *Statistical Methods in Medical Research*, 28(8), 2326–2347. doi:10.1177/0962280218773115

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(treatment = c("Control", "Experimental"), rate = c(0.5, 0.3))
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.1, 0.05), duration = c(100, 100)
)
sim <- nb_sim(enroll_rate, fail_rate, dropout_rate, max_followup = 2, n = 20)
head(sim)
```

nb_sim_seasonal *Simulate recurrent events with seasonal rates*

Description

Simulates recurrent events where event rates depend on the season.

Usage

```
nb_sim_seasonal(
  enroll_rate,
  fail_rate,
  dropout_rate = NULL,
  max_followup = NULL,
  randomization_start_date = NULL,
  n = NULL,
  block = c(rep("Control", 2), rep("Experimental", 2))
)
```

Arguments

enroll_rate	A data frame with columns rate and duration.
fail_rate	A data frame with columns treatment, season, rate, and optionally dispersion. Seasons should be "Spring", "Summer", "Fall", "Winter".
dropout_rate	A data frame with columns treatment, rate, duration.
max_followup	Numeric. Max follow-up duration (years).
randomization_start_date	Date. Start of randomization.
n	Integer. Total sample size.
block	Character vector for block randomization.

Value

A data frame of class nb_sim_seasonal with columns: id, treatment, season, enroll_time, start, end, event, calendar_start, calendar_end. Rows represent intervals of risk or events. event=1 indicates an event at end. event=0 indicates censoring or end of a seasonal interval at end.

Examples

```
enroll_rate <- data.frame(rate = 20 / (5 / 12), duration = 5 / 12)
fail_rate <- data.frame(
  treatment = rep(c("Control", "Experimental"), each = 4),
  season = rep(c("Winter", "Spring", "Summer", "Fall"), times = 2),
  rate = c(0.6, 0.5, 0.4, 0.5, 0.4, 0.3, 0.2, 0.3)
)
sim <- nb_sim_seasonal(
```

```
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  max_followup = 1,
  randomization_start_date = as.Date("2020-01-01"),
  n = 20
)
head(sim)
```

`print.gsNBsummary` *Print method for gsNBsummary objects*

Description

Print method for gsNBsummary objects

Usage

```
## S3 method for class 'gsNBsummary'
print(x, ...)
```

Arguments

`x`	An object of class `gsNBsummary`.
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object.

Examples

```
nb_ss <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.9,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
gs_design <- gsNBCalendar(nb_ss, k = 3, analysis_times = c(12, 18, 24))
s <- summary(gs_design)
print(s)
```

print.sample_size_nbinom_result

Print method for sample_size_nbinom_result objects

Description

Prints a concise summary of the sample size calculation results.

Usage

```
## S3 method for class 'sample_size_nbinom_result'
print(x, ...)
```

Arguments

x	An object of class sample_size_nbinom_result.
...	Additional arguments (currently ignored).

Value

Invisibly returns the input object.

Examples

```
x <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.8,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
print(x)
```

print.sample_size_nbinom_summary

Print method for sample_size_nbinom_summary objects

Description

Print method for sample_size_nbinom_summary objects

Usage

```
## S3 method for class 'sample_size_nbinom_summary'
print(x, ...)
```

Arguments

x	An object of class sample_size_nbinom_summary.
...	Additional arguments (currently ignored).

Value

Invisibly returns the input object.

Examples

```
x <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.8,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
s <- summary(x)
print(s)
```

---

sample_size_nbinom *Sample size calculation for negative binomial distribution*

Description

Computes the sample size for comparing two treatment groups assuming a negative binomial distribution for the outcome.

Usage

```
sample_size_nbinom(
  lambda1,
  lambda2,
  dispersion,
  power = NULL,
  alpha = 0.025,
  sided = 1,
  ratio = 1,
  rr0 = 1,
  accrual_rate,
  accrual_duration,
  trial_duration,
  dropout_rate = 0,
  max_followup = NULL,
  event_gap = NULL
)
```

Arguments

`lambda1`	Rate in group 1 (control).
`lambda2`	Rate in group 2 (treatment).
`dispersion`	Dispersion parameter k such that $\mathrm{Var}(Y) = \mu + k\mu^2$. Note that this is equivalent to `1/size` in R's `stats::rnbinom()` parameterization.
`power`	Power of the test (1 - beta). Default is 0.9.
`alpha`	Significance level. Default is 0.025.
`sided`	One-sided or two-sided test. 1 for one-sided, 2 for two-sided. Default is 1.
`ratio`	Allocation ratio n2/n1. Default is 1.
`rr0`	Rate ratio under the null hypothesis (lambda2/lambda1). Default is 1 (superiority). For non-inferiority, use a value > 1 (e.g., 1.1). For super-superiority, use a value < 1 (e.g., 0.8).
`accrual_rate`	Vector of accrual rates (patients per unit time).
`accrual_duration`	Vector of durations for each accrual rate. Must be same length as `accrual_rate`.
`trial_duration`	Total planned duration of the trial. If `trial_duration` is less than the sum of `accrual_duration`, accrual is truncated at `trial_duration`.
`dropout_rate`	Dropout rate (hazard rate). Default is 0. Can be a vector of length 2.
`max_followup`	Maximum follow-up time for any patient. Default is NULL (infinite).
`event_gap`	Gap duration after each event during which no new events are counted. Default is NULL (no gap). If provided, the effective event rate is reduced.

Value

An object of class `sample_size_nbinom_result`, which is a list containing:

inputs Named list of the original function arguments.

n1 Sample size for group 1

n2 Sample size for group 2

n_total Total sample size

alpha Significance level

sided One-sided or two-sided test

power Power of the test

exposure Average exposure time used in calculation (calendar time). Vector of length 2.

exposure_at_risk_n1 Average at-risk exposure time for group 1 (accounts for event gap)

exposure_at_risk_n2 Average at-risk exposure time for group 2 (accounts for event gap)

events_n1 Expected number of events in group 1

events_n2 Expected number of events in group 2

total_events Total expected number of events

variance Variance of the log rate ratio

accrual_rate Accrual rate used in calculation

accrual_duration Accrual duration used in calculation

References

Zhu, H., & Lakkis, H. (2014). Sample size calculation for comparing two negative binomial rates. *Statistics in Medicine*, 33(3), 376–387. doi:10.1002/sim.5947

Friede, T., & Schmidli, H. (2010). Blinded sample size reestimation with negative binomial counts in superiority and non-inferiority trials. *Methods of Information in Medicine*, 49(06), 618–624. doi:10.3414/ME09020060

Mütze, T., Glimm, E., Schmidli, H., & Friede, T. (2019). Group sequential designs for negative binomial outcomes. *Statistical Methods in Medical Research*, 28(8), 2326–2347. doi:10.1177/0962280218773115

See Also

`vignette("sample-size-nbinom", package = "gsDesignNB")` for a detailed explanation of the methodology.

Examples

```
# Calculate sample size for lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1
# with fixed recruitment of 10/month for 20 months, 24 month trial duration
x <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.8,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
class(x)
summary(x)

# With piecewise accrual
# 5 patients/month for 3 months, then 10 patients/month for 3 months
# Trial ends at month 12.
x2 <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.8,
  accrual_rate = c(5, 10), accrual_duration = c(3, 3),
  trial_duration = 12
)
summary(x2)
```

`sim_gs_nbinom` *Simulate group sequential clinical trial for negative binomial outcomes*

Description

Simulates multiple replicates of a group sequential clinical trial with negative binomial outcomes, performing interim analyses at specified calendar times.

Usage

```
sim_gs_nbinom(
  n_sims,
  enroll_rate,
  fail_rate,
  dropout_rate = NULL,
  max_followup,
  event_gap = 0,
  analysis_times = NULL,
  n_target = NULL,
  design = NULL,
  data_cut = cut_data_by_date,
  cuts = NULL
)
```

Arguments

n_sims	Number of simulations to run.
enroll_rate	Enrollment rates (data frame with rate and duration).
fail_rate	Failure rates (data frame with treatment, rate, dispersion).
dropout_rate	Dropout rates (data frame with treatment, rate, duration).
max_followup	Maximum follow-up time.
event_gap	Event gap duration.
analysis_times	Vector of calendar times for interim and final analyses. Optional if cuts is provided.
n_target	Total sample size to enroll (optional, if not defined by enroll_rate).
design	An object of class gsNB or sample_size_nbinom_result. Used to extract planning parameters (lambda1, lambda2, ratio) for blinded information estimation.
data_cut	Function to cut data for analysis. Defaults to cut_data_by_date(). The function must accept sim_data, cut_date, and event_gap as arguments.
cuts	A list of cutting criteria for each analysis. Each element of the list should be a list of arguments for get_cut_date() (e.g., planned_calendar, target_events, target_info). If provided, analysis_times is ignored (or used as a fallback if planned_calendar is missing in a cut).

Value

A data frame containing simulation results for each analysis of each trial. Columns include:

sim Simulation ID

analysis Analysis index

analysis_time Calendar time of analysis

n_enrolled Number of subjects enrolled

n_ctrl Number of subjects in control group

n_exp Number of subjects in experimental group

events_total Total events observed

events_ctrl Events in control group

events_exp Events in experimental group

exposure_at_risk_ctrl Exposure at risk in control group (adjusted for event gaps)

exposure_at_risk_exp Exposure at risk in experimental group (adjusted for event gaps)

exposure_total_ctrl Total exposure in control group (calendar follow-up)

exposure_total_exp Total exposure in experimental group (calendar follow-up)

z_stat Z-statistic from the Wald test (positive favors experimental if rate ratio < 1)

estimate Estimated log rate ratio from the model

se Standard error of the estimate

method_used Method used for inference ("nb" or "poisson")

dispersion Estimated dispersion parameter from the model

blinded_info Estimated blinded statistical information (ML)

unblinded_info Observed unblinded statistical information (ML)

info_unblinded_ml Observed unblinded statistical information (ML)

info_blinded_ml Estimated blinded statistical information (ML)

info_unblinded_mom Observed unblinded statistical information (Method of Moments)

info_blinded_mom Estimated blinded statistical information (Method of Moments)

Examples

```
set.seed(123)
enroll_rate <- data.frame(rate = 10, duration = 3)
fail_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.6, 0.4),
  dispersion = 0.2
)
dropout_rate <- data.frame(
  treatment = c("Control", "Experimental"),
  rate = c(0.05, 0.05),
  duration = c(6, 6)
)
design <- sample_size_nbinom(
  lambda1 = 0.6, lambda2 = 0.4, dispersion = 0.2, power = 0.8,
  accrual_rate = enroll_rate$rate, accrual_duration = enroll_rate$duration,
  trial_duration = 6
)
cuts <- list(
  list(planned_calendar = 2),
  list(planned_calendar = 4)
)
sim_results <- sim_gs_nbinom(
  n_sims = 2,
```

```
  enroll_rate = enroll_rate,
  fail_rate = fail_rate,
  dropout_rate = dropout_rate,
  max_followup = 4,
  n_target = 30,
  design = design,
  cuts = cuts
)
head(sim_results)
```

`summarize_gs_sim` *Summarize group sequential simulation results*

Description

Provides a summary of the operating characteristics of the group sequential design based on simulation results.

Usage

```
summarize_gs_sim(x)
```

Arguments

`x`	A data frame returned by `check_gs_bound()` (or `sim_gs_nbinom()` if bounds are manually checked). Must contain columns `cross_upper`, `cross_lower`.

Value

A list containing:

n_sim Number of simulations

power Overall power (probability of crossing upper bound)

futility Overall futility rate (probability of crossing lower bound and not upper)

analysis_summary Data frame with per-analysis statistics (events, crossings)

Examples

```
design <- gsDesign::gsDesign(k = 2, n.fix = 80, test.type = 2, timing = c(0.5, 1))
sim_df <- data.frame(
  sim = c(1, 1, 2, 2),
  analysis = c(1, 2, 1, 2),
  z_stat = c(2.4, NA, -0.5, 1.9),
  blinded_info = c(40, 80, 40, 80),
  unblinded_info = c(40, 80, 40, 80),
  n_enrolled = c(30, 60, 30, 60),
  events_total = c(12, 25, 10, 22)
)
bounds_checked <- check_gs_bound(sim_df, design)
summarize_gs_sim(bounds_checked)
```

summary.gsNB *Summary for gsNB objects*

Description

Provides a textual summary of a group sequential design for negative binomial outcomes, similar to the summary provided by gsDesign::gsDesign(). For tabular output, use gsDesign::gsBoundSummary() directly on the gsNB object.

Usage

```
## S3 method for class 'gsNB'
summary(object, ...)
```

Arguments

object	An object of class gsNB.
...	Additional arguments (currently ignored).

Value

A character string summarizing the design (invisibly). The summary is also printed to the console.

Examples

```
nb_ss <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.9,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
gs_design <- gsNBCalendar(nb_ss, k = 3, analysis_times = c(12, 18, 24))
summary(gs_design)

# For tabular bounds summary, use gsBoundSummary() directly:
gsBoundSummary(gs_design)
```

summary.sample_size_nbinom_result
Summary for sample_size_nbinom_result objects

Description

Provides a textual summary of the sample size calculation for negative binomial outcomes, similar to the summary for gsNB objects.

Usage

```
## S3 method for class 'sample_size_nbinom_result'
summary(object, ...)
```

Arguments

object	An object of class sample_size_nbinom_result.
...	Additional arguments (currently ignored).

Value

A character string summarizing the design (invisibly). The summary is also printed to the console.

Examples

```
x <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.8,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
class(x)
summary(x)
```

---

toInteger *Convert group sequential design to integer sample sizes*

Description

Generic function to round sample sizes in a group sequential design to integers. This extends the gsDesign::toInteger() function from the gsDesign package to work with gsNB objects.

Usage

```
toInteger(x, ...)

## S3 method for class 'gsDesign'
toInteger(x, ratio = x$ratio, roundUpFinal = TRUE, ...)

## S3 method for class 'gsNB'
toInteger(x, ratio = x$nb_design$inputs$ratio, roundUpFinal = TRUE, ...)
```

Arguments

x	An object of class gsNB or gsDesign.
...	Additional arguments passed to methods.
ratio	Randomization ratio (n2/n1). If an integer is provided, rounding is done to a multiple of ratio + 1. If ratio < 1 and 1/ratio is an integer (e.g., 1:2 allocation, ratio = 0.5), rounding is done to a multiple of 1/ratio + 1. Default uses the ratio from the original design.

roundUpFinal If TRUE (default), the final sample size is rounded up to ensure the target is met. If FALSE, rounding is to the nearest integer.

Details

This function rounds sample sizes at each analysis to integers while maintaining the randomization ratio and ensuring monotonically increasing sample sizes across analyses. Only the final analysis sample size is rounded to an integer; interim sample sizes remain as expected (non-integer) values based on the information fraction.

When analysis_times were provided to gsNBCalendar(), the statistical information (n.I) is re-computed at each analysis time based on the new sample size and expected exposures.

Value

An object of the same class as input with integer sample sizes.

Methods (by class)

- toInteger(gsDesign): Method for gsDesign objects (calls gsDesign::toInteger()).
- toInteger(gsNB): Method for gsNB objects.

  Rounds sample sizes in a group sequential negative binomial design to integers, respecting the randomization ratio.

Examples

```
nb_ss <- sample_size_nbinom(
  lambda1 = 0.5, lambda2 = 0.3, dispersion = 0.1, power = 0.9,
  accrual_rate = 10, accrual_duration = 20, trial_duration = 24
)
gs_design <- gsNBCalendar(nb_ss, k = 3, analysis_times = c(12, 18, 24))
gs_integer <- toInteger(gs_design)
```

---

unblinded_ssr *Unblinded sample size re-estimation for recurrent events*

Description

Estimates the event rates and dispersion from unblinded interim data and calculates the required sample size to maintain power, assuming the planned treatment effect holds (or using the observed control rate).

Usage

```
unblinded_ssr(
  data,
  ratio = 1,
  lambda1_planning,
  lambda2_planning,
  rr0 = 1,
  power = 0.8,
  alpha = 0.025,
  accrual_rate,
  accrual_duration,
  trial_duration,
  dropout_rate = 0,
  max_followup = NULL,
  event_gap = NULL
)
```

Arguments

`data`	A data frame containing the unblinded interim data. Must include columns `events` (number of events), `tte` (total exposure/follow-up time), and `treatment` (treatment group identifier, e.g., 1 for control, 2 for experimental). This is typically the output of `cut_data_by_date()`.
`ratio`	Planned allocation ratio (experimental / control). Default is 1.
`lambda1_planning`	Planned event rate for the control group used in original calculation.
`lambda2_planning`	Planned event rate for the experimental group used in original calculation.
`rr0`	Rate ratio under the null hypothesis (lambda2/lambda1). Default is 1.
`power`	Target power (1 - beta). Default is 0.8.
`alpha`	One-sided significance level. Default is 0.025.
`accrual_rate`	Vector of accrual rates (patients per unit time).
`accrual_duration`	Vector of durations for each accrual rate. Must be same length as `accrual_rate`.
`trial_duration`	Total planned duration of the trial.
`dropout_rate`	Dropout rate (hazard rate). Default is 0.
`max_followup`	Maximum follow-up time for any patient. Default is NULL (infinite).
`event_gap`	Gap duration after each event during which no new events are counted. Default is NULL (no gap).

Value

A list containing:

n_total_unblinded Re-estimated total sample size using unblinded estimates.

dispersion_unblinded Estimated dispersion parameter (k) from unblinded data.

lambda1_unblinded Estimated control event rate from unblinded data.

lambda2_unblinded Estimated experimental event rate from unblinded data.

info_fraction Estimated information fraction at interim (unblinded information / target information).

unblinded_info Estimated statistical information from the unblinded interim data.

target_info Target statistical information required for the planned power.

Examples

```
interim <- data.frame(
  events = c(1, 2, 1, 3),
  tte = c(0.8, 1.0, 1.2, 0.9),
  treatment = c("Control", "Control", "Experimental", "Experimental")
)
unblinded_ssr(
  interim,
  ratio = 1,
  lambda1_planning = 0.5,
  lambda2_planning = 0.3,
  power = 0.8,
  alpha = 0.025,
  accrual_rate = 10,
  accrual_duration = 12,
  trial_duration = 18
)
```

Index