Package: confidenceSim (via r-universe)

May 23, 2026

Title Highly Customizable, Parallelized Simulations of Frequentist Confidence Clinical Trials

Version 0.1.0

Description Simulate one or many frequentist confidence clinical trials based on a specified set of parameters. From a two-arm, single-stage trial to a perpetually run Adaptive Platform Trial, this package offers vast flexibility to customize your trial and observe operational characterisitics over thousands of instances.

License MIT + file LICENSE

Encoding UTF-8

Language es

RoxygenNote 7.3.2

Imports confidenceCurves (>= 0.2.0), genodds (>= 1.1.2), rpact (>= 4.0.0)

Depends R (>= 4.3)

LazyData true

Suggests dplyr (>= 1.1.4), knitr, parallel (>= 4.3.2), pbapply (>= 1.7.2), plyr (>= 1.8.9), rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Freda Werdiger [aut, cre] (ORCID: <https://orcid.org/0000-0002-5535-7117>)

Maintainer Freda Werdiger <freda.werdiger@unimelb.edu.au>

Repository https://cran.r-universe.dev

Date/Publication 2025-10-25 12:20:07 UTC

RemoteUrl https://github.com/cran/confidenceSim

RemoteRef HEAD

RemoteSha 4be9583855f067b1a6ea40ff04b83159f4b14f2a

Contents

`datlist` *Simulated trial data*

Description

Simulated trial data list `datlist` that is generated by `singleTrial.R` and contains all the necessary information to perform confidence analysis.

Usage

```
data(datlist)
```

Format

An object of class `"list"`

subjid Subject ID from 1 to maximum sample size

arm Treatment arm allocation for each subject

dat Response for each subject (0 or 1)

arrival.day Arrival time (days) for each subject throughout trial duration

obstime Obveration time (days) for each subject; when response is observed

References

This data set was created for the confidenceSim package.

Examples

```
data(datlist)
head(datlist)
```

getAccrual *Get Accrual*

Description

Generate patient accrual with Poisson distribution.

Usage

```
getAccrual(
  numsubjects,
  ppm,
  follow.up = 0,
  cont.recruit = FALSE,
  perpetual = FALSE
)
```

Arguments

numsubjects	Maximum sample size. If perpetual is TRUE, a new maximum sample size is returned.
ppm	Patients accrued per month, as an array. Length of array is the number of months in the trial.
follow.up	Follow-up period in months.
cont.recruit	Whether to continue recruitment while waiting for follow up (TRUE) of not (FALSE).
perpetual	Whether to run trial perpetually (TRUE) or not (FALSE).

Value

Vector size of number of patients you need to get 'numsubjects' followup with values representing month of accrual.

Examples

```
ppm <- rep(15, 300)
monthin <- getAccrual(1000, ppm, 0)
# monthin is of length 1000.
```

getBlockedArm *Get Blocked Arm*

Description

Randomize patients to arms using blocked randomization.

Usage

```
getBlockedArm(numsubjects, num.per.block, prob = NULL)
```

Arguments

numsubjects	Number of subjects to randomize.
num.per.block	Number from each arm per block. Block size is 'sum(num.per.block)'.
prob	Probability of randomization to each arm. Default assumes equal probability.

Details

To balance covariates, each block gets an equal distribution of treatment arms to remove the effect that could come from the block characteristics (e.g. covariates). If not balancing covariates, patients will be randomized according to ratios.

Value

Return vector of arm allocations.

Examples

```
arm <- getBlockedArm(500, c(1,1))
```

getBoundsFromConfidence *Get Critical Bounds from Confidence Thresholds*

Description

Derive traditional frequentist critical values from frequentist confidence thresholds for confidence in treatment benefit.

Usage

```
getBoundsFromConfidence(
  num.treat.arms = 2,
  conf.lower = 0.01,
  conf.upper = 0.99,
  p.sided = 1
)
```

Arguments

num.treat.arms	Number of treatment arms (excludes control). Default is 2.
conf.lower	Confidence in treatment benefit boundary for inferiority i.e. stop for inferiority if confidence in benefit is below this. Default is 0.01.
conf.upper	Confidence in treatment benefit boundary for efficacy i.e. stop for efficacy if confidence in benefit is above this. Default is 0.99.
p.sided	Sidedness of statistical test, 1 (one-sided) and 2 (two-sided). Default is 1.

Details

During a confidence trial, efficacy and inferiority is determined by the level of confidence in treatment benefit. Efficacy is declared if this confidence level exceeds a pre-specified boundary, and inferiority is declared if this confidence levels falls below a second pre-specified valye. Given confidence-based thresholds for efficacy and inferiority, and the sidedness of the test, this function returns the traditional frequentist p-value.

Value

List of values:

- conf.lower: confidence in treatment benefit lower bound
- z.score.lower: critical value corresponding to lower confidence bound
- p.value.lower: p-value correponding to lower confidence bound
- conf.upper: confidence in treatment benefit upper bound
- z.score.upper: critical value corresponding to upper confidence bound
- p.value.upper: p-value corresponding to upper confidence bound
- p.value: sidedness of test

Examples

```
# Running the function on default values
bounds <- getBoundsFromConfidence()

# to make adjustments for multiple arms
bounds <- getBoundsFromConfidence(num.treat.arms = 3)
```

---

getConfidenceFromBounds

Get Confidence Levels from Group Sequential Bounds

Description

Derive confidence-based decision thresholds for efficacy and inferiority from a group sequential design.

Usage

```
getConfidenceFromBounds(design)
```

Arguments

design	TrialDesign object generated from 'getGSDesign'.

Value

List of values:

- critical.values: Critical z-levels at each stage
- alpha.spending.one.sided: One-sided alpha critical values for each stage
- alpha.spending.cumulative: One-sided alpha accumulated by each stage
- confidence.threshold.efficacy: Upper bounds for confidence i.e. declare efficacy if exceeded
- confidence.threshold.inferiority: Lower bounds for confidence i.e. declare inferiority if below

See Also

getGSDesign()

Examples

```
# confidence bounds for a 6-stage trial with a maximum sample size of 1000
bounds <- getConfidenceFromBounds(getGSDesign(looks=seq(500, 1000, 100)))
```

---

getCurrentData *Get Current Data*

Description

Get the data available at the time of the current analysis.

Usage

```
getCurrentData(datlist, looktime, n, as.followup = TRUE)
```

Arguments

datlist	The entire data list generated at the start of the simulation.
looktime	The time of the current analysis point.
n	The number of subjects corresponding to this analysis point (n.at.look).
as.followup	If TRUE, the true looktime is when all *n* patient reach follow-up. If FALSE, looktime remains the time when the *n*th patient is *enrolled*. Default is TRUE.

Value

Returns a subset of the data to use in interim analysis.

- subjid: Subject ID from 1 to maximum sample size
- arm: Treatment arm allocation for each subject
- dat: Response for each subject (0 or 1)
- arrival.day: Arrival time (days) for each subject throughout trial duration

Examples

```
# using included data set
data(datlist)
# This example uses the default parlist parameters
looks <- seq(500,1000,100)

# get the data available at the first interim analysis
n.at.look <- looks[1]
looktime.interim <- datlist$arrival.day[n.at.look]
currdatlist.interim <- getCurrentData(datlist, looktime.interim, n.at.look, as.followup=TRUE)
# currdatlist.interim will have a new field `KNOWN`
#3 which indicates if a patients response is known (TRUE) or not (FALSE)
```

---

getDataBin *Generate binary data*

Description

Given an arm allocation and response rates, this function generates a binary response.

Usage

```
getDataBin(arm, resprate)
```

Arguments

arm	Arm allocation for a single patient.Expects number in *1,2,...,n* where *n* the number of treatment arms including control.
resprate	Response rates for each arm. Expects a vector of probabilities of length *n* with the first corresponding to response rate of the control arm.

Value

Returns a binary value corresponding to patient response.

Examples

```
response <- getDataBin(1, c(0.5, 0.7))
```

getDataCont *Get Continuous Data*

Description

Given an arm allocation and response rates, this function generates response from a given distribution.

Usage

```
getDataCont(arm, resprate, dist = "norm")
```

Arguments

arm	Arm allocation for a single patient. Expects number in *1,2,...,n* where *n* is the number of treatment arms including control.
resprate	Response rates for each arm. Expects a list of n lists with the first list containing median and standard deviation paramaterizing control response. Expects `c(mean, sd)` for each arm.
dist	Type of distribution. Default is normal (`norm`).

Value

Returns a continuous value corresponding to patient response.

Examples

```
response <- getDataCont(1, list(control = c(0,1), treatment = c(0.5,1)), dist='norm')
```

getDataOrd *Generate ordinal data*

Description

Given an arm allocation and response rates, this function generates response on an ordinal scale.

Usage

```
getDataOrd(arm, resprate)
```

Arguments

arm	Arm allocation for a single patient. Expects number in *1,2,...,n* where *n* is the number of treatment arms including control.
resprate	Response rates for each arm. Expects a list of *n* lists with the first list containing probabilities for response level which correspond to the control arm.

Value

Returns an ordinal value corresponding to patient response.

Examples

```
# for a three-point ordinal scale
response <- getDataOrd(1, list(control = c(0.3, 0.5, 0.7), treatment = (c(0.5, 0.3, 0.2))))
```

`getGSDesign` *Get Group Sequential Design*

Description

Generate boundaries for a group sequential design using a one-sided test

Usage

```
getGSDesign(info.rates = NULL, looks = NULL, as.type = "asOF")
```

Arguments

`info.rates`	Analysis times expressed as rate of information accrual. Expects a vector with the last item representing the final analysis and equal to 1. For example, information rates for two-stage trial with interim analysis half way through is c(0.5, 1). One of two options for expressing analysis times. Either 'info.rates' or 'looks' must be specified.
`looks`	Analysis times expressed by number of patients accrued at each point. Expects a vector with the last item being equal to the maximum sample size. For example. looks for a three stage trial with maximum sample size of 300 and analysis planned every 100 patients is c(100, 100, 100). One of two options for expressing analysis times. Either 'info.rates' or 'looks' must be specified.
`as.type`	Time of alpha spending function to use. Options are as outlined by 'rpact'. Default is 'asOF' \(O'Brien-Fleming-type\).

Details

To generate confidence-based thresholds, we are interested in a one-sided test and alpha is set at 0.025. To generate the stopping thresholds, specify either looks or information rates, and the alpha spending function if difference from O-Brien-Fleming-type.

Value

Returns an 'rpact' TrialDesign object.

See Also

`rpact::getDesignGroupSequential()`

Examples

```
# calculate critical values for a two-stage trial with an interim analysis half-way through
# Use Pocock-type alpha spending
design <- getGSDesign(info.rates = c(0.5, 1), as.type = 'asP')
critical.stagewise.alpha.levels <- design$stageLevels

# calculate values for a 6-stage trial with a maximum sample size of 1000
# interim analysis begins at 500 patients accrued and continues every 100 patients after
design <- getGSDesign(looks = seq(500, 1000, 100))
```

---

getparlist *Get Parameter List*

Description

Generate a parameter list to generate a frequentist confidence trial

Usage

```
getparlist(
  looks = seq(500, 1000, 100),
  nmax = NULL,
  perpetual = FALSE,
  alloc.ratio = c(1, 1),
  num.per.block = c(1, 1),
  final.visit = 0,
  as.type = "asOF",
  alpha = 0.05,
  multiarm.mode = "CONFIDENCE-BASED",
  lmb.threshold = 0.1,
  lmb.conf.thresh = 0.9,
  outcome.type = "BINARY",
  estimator.type = "odds ratio",
  resprate = c(0.3, 0.5),
  ppm = rep(15, 300),
  special = NULL
)
```

Arguments

looks	Vector of analysis times expressed by either number of patients accrued at each point or by rate of information accumulated. If the former, last item should be the maximum sample size. If the latter, last item is 1. Expects a vector with length equal to the total number of total looks.
nmax	Maximum sample size, specified if information rates are used for 'looks'.

`perpetual` Whether to run the trial perpetually (TRUE) or not (FALSE). If TRUE, new treatment arms will be added when treatment arms are dropped, until there are no more arms left. All treatments are included via the 'resprate' parameter. Default is FALSE.

`alloc.ratio` Allocation ratios for study arms relative to each other. Expects vector with length equal to number of arms including control. First number corresponds to control ratio.

`num.per.block` Number from each arm per block, for blocked randomization to balance covariates. Block size is 'sum(num.per.block)'. If a single number is provided, it will be assume to apply to each arm.

`final.visit` The number of days after intervention when the response information becomes available. Default assumes immediate follow-up (0).

`as.type` The type of alpha spending function to use in group sequential design. Default is 'asOF', O'Brien-Fleming-type.

`alpha` The alpha threshold to apply to each pairwise comparison to control in the final analysis. Used together with the 'MONITOR FUTILITY', when an alpha spending function is not needed. Default is 0.05, assuming a two-sided test.

`multiarm.mode` For multiple treatment arms, describes how arms are evaluated at each stage:

- "CONFIDENCE-BASED"(default): Evaluate arms against confidence-based rules
- "DROP WORST": Drop the worst performing arm, and carry the remaining promising arms
- "SELECT BEST": Select the best performing arm to carry forward, drop the rest
- "ALL PROMISING": Carry forward all promising arms
- "MONITOR FUTILITY": Only monitor for futility

`lmb.threshold` Defined threshold for meaningful benefit. The direction of benefit/lacks benefit depends on the data and outcome, and whether lower or higher is better. For ordinal data, lower is better and anything greater than lmb.threshold lacks meaningful benefit. In that case we use a `genodds` estimator and `lmb.threshold` should be below 1. For binary and continous data, higher is better and `lmb.threshold` should reflect that. If the treatment effect is a ratio, it will be later converted to the logarithmic scale for confidence analysis.

`lmb.conf.thresh` Confidence threshold for futility. If confidence in lack of meaningful benefit (LMB) is greater than this for a given treatment arm, the arm may be dropped. Default is 0.9.

`outcome.type` Type of primary outcome: "CONTINUOUS", "ORDINAL", or "BINARY".

`estimator.type` Type of estimator for binary data: "odds ratio", "risk diff", "risk ratio". Default is odds ratio. For ordinal data, a generalised odds ratio 'genodds' is used. For continuous data, a difference of means is used.

`resprate` The response rates for control and treatment. For binary and continuous data, expects a vector which one number for each arm. For ordinal data, expects a list of lists with the with list corresponding to control. If running perpetually, include all treatments here, including those that will not initially be in the trial.

ppm	Patients per month.While the maximum sample size for a non-perpetual trial is derived from 'looks', in a perpetually setting the trial will continue to go so long as there are new treatments to add, and patients are still accruing according to the length of ppm.
special	Any information wishing to pass to the tag that will be added to the results dataframe under the 'misc' column.

Value

A parameter list used to generate a trial.

Examples

```
# two-arm six-stage trial (PRESTO-REACH) with binary outcome measure

parlist <- getparlist(
looks=seq(500,1000,100),
perpetual=FALSE,
alloc.ratio=c(1,1),
num.per.block=c(1,1),
final.visit=0,
as.type="asOF",
multiarm.mode="CONFIDENCE-BASED",
lmb.threshold=0.95,
lmb.conf.thresh=0.9,
outcome.type='BINARY',
estimator.type='odds ratio',
resprate=c(0.3,0.5),
ppm=rep(15, 300))

# two-arm three-stage trial with 16-point ordinal outcome

resprate <- list(
ctrl = rep(1/16, 16),
trmt=c(
0.08119658, 0.07802130, 0.07502870,0.07220504, 0.06953783,0.06701574,
0.06462841, 0.06236641, 0.06022113,0.05818467, 0.05624978, 0.05440984,
0.05265872, 0.05099079,0.04940088, 0.04788419)
)

# create a list of input parameters

inputs <- list(
 lmb.threshold = 1.10,
 as.type = 'asOF',
 outcome.type = "ORDINAL",
 multiarm.mode='CONFIDENCE-BASED',
 num.per.block = c(1,1),
 final.visit = 180,
 ppm = rep(20, 300),
 perpetual=FALSE,
 resprate=resprate,
```

```
looks=c(500,1000,1500)
)
# pass parameters in through "inputs"
parlist <- do.call("getparlist", inputs)
```

---

getSuffStats *Get Sufficient Statistics*

Description

Get sufficient statistics from trial data necessary to perform primary analysis

Usage

```
getSuffStats(datlist)
```

Arguments

datlist	The current data list at the point of analysis generated from getCurrentData. The list must have a KNOWN field.

Details

Given a data list, this checks if the responses are binary (two types of responses), ordinal (more than 2 or less than 30 different response types) or continuous (other). If continuous, mean and standard deviations are returned. This code is not necessarily used in singleTrial since the perpetual functionality was introduced as there are additional methods to retrieve the statistics necessary to perform the analysis that take into account the possibility that arms have been dropped or added.

Value

List of sufficient statistics

- num.enrolled
- num.known
- num.uknown
- num.resp
- num.fail
- resprate
- formattedrate

Examples

```
# load data set
data(datlist)
looks <- seq(500,1000,100)
# first interim analysis
n.at.look = looks[1]
looktime.interim = datlist$arrival.day[n.at.look]
currdatlist.interim <- getCurrentData(datlist, looktime.interim, n.at.look, as.followup=TRUE)
suffStats  <- getSuffStats(currdatlist.interim)
```

`inputs` *Example trial input parameters*

Description

List of parameter that contains all the necessary information to generate a trial.

Usage

```
data(inputs)
```

Format

An object of class `"list"`

Parameter list used to generate a trial, in this case a 2-arm 6-stage trial with options to stop early for efficacy, inferiority or futility based on confidence thresholds

References

This data set was created for the confidenceSim package.

See Also

`getparlist()`

Examples

```
data(inputs)
print(inputs)
```

runSingleTrial *Frequentist Confidence-Adaptive Trial Simulation*

Description

Simulates a group sequential clinical trial whose result is evaluated via frequentist confidence analysis.

Usage

```
runSingleTrial(
  sim.no = 0,
  inputs = NULL,
  save.plot = FALSE,
  save.text = TRUE,
  show = "BENEFIT",
  directory = "",
  reproduce = FALSE,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

sim.no	Simulation number, when running mutiple simulations of a trial.
inputs	A list of items fed to the function which parameterize the trial to be simulated. An example parameter list can be loaded using `data(inputs)`.
save.plot	Whether or not to save confidence curve plot with the result.TRUE (yes) or FALSE (no). When running multiple simulations, `FALSE` is recommended. If `TRUE`, files will be saved to the specified directory. The filename is automatically generated according to trial settings. Default is FALSE. Passed to `makeConfidenceCurves`.
save.text	Whether or not to save results to directory. Default is TRUE.
show	If saving confidence curves, what to show on the confidence density plot. Options are "BENEFIT" (default), "LMB" (lack of meaningful benefit), "MB" (meaningful benefit) or "EQUIV" (equivalence). Passed to `makeConfidenceCurves`.
directory	Working directory. Used to save Random State, and trial results. A subdirectory is created based on the current node to allow for parallel computing across multiple nodes. Random State is checkpointed throughout the code and saved in the subdirectory 'directory/node/'. Results are saved in the same places as the Random States. If `save.text == FALSE` nothing is saved to `directory`.
reproduce	To reproduce a result from saved Random States. If setting as TRUE, make sure the directory parameter points to the location (the node subdirectory) where the Random States are saved. The results will be saved to this directory. If set as FALSE (default), results and Random States are saved to the node subdirectory.

verbose	Whether to print out text (TRUE) or not (FALSE). Useful to observe the trial process and decision-making while the simulation is running. Not recommended if running a high number of simulations.
seed	Option to set the seed. Default is NULL.

Details

Run simulations of a confidence-based adaptive clinical trial for any number of arms and stages. At each analysis point, the confidence in treatment benefit and futility is evaluated and arms may be dropped or continued based on the trial settings. The trial may also be run perpetually, with new treatment arms being added once arms are dropped, as an adaptive platform trial. The confidence-based thresholds are derived using an alpha spending function, or specified with a fixed alpha.

Value

Object where each line is the result of confidence analysis for a given arm, at a given stage. The number of lines is the number experimental treatments * number of stages, e.g., A two-arm-two-stage trial returns a two-line object. However, if the trial is stopped after the first stage, only one line is returned.

Attributes in output object:

- sim.no: simulation number
- arm: arm number starting from 2 as 1 is control
- interim.arm: stage number of this arm (will differ from interim.total if arm was added later)
- interim.total: interim number for trial
- mean: point estimate
- standard.error: standard error associated with point estimate
- resp.ctrl: for binary data, resulting control response rate
- resp.trmt: for binary data, resulting treatment response rate
- conf.benefit: confidence in treatment benefit
- conf.lack.meaningful.benefit: confidence in lack of meaningful benefit
- action: Decision taken at this analysis point e.g. stop early, continue
- N.looks: Number of looks (analysis points) in this trial design
- misc: Information passed into getparlist function by the 'special' parameter. Can be anything of interest
- N.arm: Number recruited to this treatment arm
- N.pair: Number of patients in this two-arm (pairwise) analysis against control
- N.known: Number of patients with a known outcome (prespecified in trial settings)
- N: Total number of patients recruited so far. Will be different from N.known if there is a follow-up period.

References

Frequentist confidence analysis is based on Marschner (2024) doi:10.1002/sim.10000.

See Also

confidenceCurves::makeConfidenceCurves()

Examples

```
# Example of input list to generate a two-arm-two-stage trial with binary outcome data

inputs <- list(
 outcome.type = "BINARY", # binary outcome data
 estimator.type = 'risk diff', # primary outcome is risk difference
 lmb.threshold = 0.1, # risk difference < 0.1 lacks meaningful benefit
 multiarm.mode='MONITOR FUTILITY', # only monitor for futility
 alpha = 0.0125, # fixed alpha threshold to determine treatment efficacy
 alloc.ratio = c(1,1), # allocation ratio
 num.per.block = c(1,1), # number per block for blocked allocation
 final.visit = 0, # time in days after which follow-up data becomes available
 ppm = rep(25, 15), # patients accrued each month for the entire trial period.
 looks = c(107, 214), # number of patients accrued at each look time, nmax = 214.
 perpetual=FALSE, # not a perpetual trial.
 resprate =  c(0.5, 0.6), # response rate for each arm
 lmb.conf.thres=0.95, # treatment arm is futility is the confidence in LMB is greater than 0.95
 special = paste0(0.5, '_', 0.6) # passing the response rates to special to add to the output
)
# run a single simulation with these settings
conf <- runSingleTrial(input=inputs, save.plot=FALSE,
save.text=FALSE, verbose=TRUE, directory = '')
```

Index