

# Package ‘BayesianPlatformDesignTimeTrend’

November 22, 2023

**Title** Simulate and Analyse Bayesian Platform Trial with Time Trend

**Version** 1.2.2

**Author** Ziyang Wang [aut, cre],  
David Woods [ctb]

**Maintainer** Ziyang Wang <zw7g20@soton.ac.uk>

**Description** Simulating the sequential multi-arm multi-stage or platform trial with Bayesian approach using the 'rstan' package, which provides the R interface for the Stan.  
This package supports fixed ratio and Bayesian adaptive randomization approaches for randomization.

Additionally, it allows for the study of time trend problems in platform trials.

There are demos available for a multi-arm multi-

stage trial with two different null scenarios, as well as for Bayesian trial cutoff screening.

The Bayesian adaptive randomisation approaches are described in:

Trippa et al. (2012) <doi:10.1200/JCO.2011.39.8420> and

Wathen et al. (2017) <doi:10.1177/1740774517692302>.

The randomisation algorithm is described in:

Zhao W <doi:10.1016/j.cct.2015.06.008>.

The analysis methods of time trend effect in platform trial are described in:

Saville et al. (2022) <doi:10.1177/17407745221112013> and

Bofill Roig et al. (2022) <doi:10.1186/s12874-022-01683-w>.

**URL** <https://github.com/ZXW834/BayesianPlatformDesignTimeTrend>

**BugReports** <https://github.com/ZXW834/BayesianPlatformDesignTimeTrend/issues>

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Biarch** true

**Depends** R (>= 4.0.0), rstan (>= 2.32.1)

**Imports** BiocManager (>= 1.30.19), boot (>= 1.3-28), doParallel (>= 1.0.17), foreach (>= 1.5.1), ggplot2 (>= 3.4.0), ggpubr (>= 0.4.0), iterators (>= 1.0.13), laGP (>= 1.5-9), lhs (>= 1.1.6), matrixStats (>= 0.61.0), methods, RColorBrewer (>= 1.1-3), Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), reshape (>= 0.8.8), rstantools (>= 2.3.1.1), stringr (>= 1.4.0)

**LinkingTo** BH ( $\geq 1.66.0$ ), Rcpp ( $\geq 0.12.0$ ), RcppEigen ( $\geq 0.3.3.3.0$ ),  
RcppParallel ( $\geq 5.0.1$ ), StanHeaders ( $\geq 2.26.0$ ), rstan ( $\geq 2.26.1$ )

**SystemRequirements** GNU make

**License** MIT + file LICENSE

**Suggests** rmarkdown, knitr

**VignetteBuilder** knitr

**LazyData** true

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-11-22 01:00:02 UTC

## R topics documented:

|   |    |
|---|----|
| BayesianPlatformDesignTimeTrend-package . . . . . | 3  |
| AdaptiveRandomisation . . . . .                   | 3  |
| alphaspending . . . . .                           | 6  |
| ARmethod . . . . .                                | 7  |
| Boundaryconstruction . . . . .                    | 9  |
| conjunctivepower_or_FWER . . . . .                | 9  |
| dataloginformd . . . . .                          | 10 |
| demo_Cutoffscreening . . . . .                    | 11 |
| demo_Cutoffscreening.GP . . . . .                 | 12 |
| demo_multiscenario . . . . .                      | 15 |
| disconjunctivepowerfunc . . . . .                 | 16 |
| GP.optim . . . . .                                | 16 |
| ibetabinomial.post . . . . .                      | 18 |
| Initializetrialparameter . . . . .                | 19 |
| intbias . . . . .                                 | 21 |
| Meanfunc . . . . .                                | 21 |
| modelinf.fun . . . . .                            | 22 |
| Nfunc . . . . .                                   | 23 |
| OPC_alt . . . . .                                 | 24 |
| OPC_null . . . . .                                | 25 |
| OPC_Trial.simulation . . . . .                    | 26 |
| optimdata_asy . . . . .                           | 26 |
| optimdata_sym . . . . .                           | 27 |
| OutputStats.initialising . . . . .                | 27 |
| perHtypeIerror_marginalpowerfunc . . . . .        | 28 |
| predictedtpIEinformd . . . . .                    | 29 |
| Randomisation.inf . . . . .                       | 29 |
| recommandloginformd . . . . .                     | 30 |
| resultrtostats . . . . .                          | 30 |
| resultrtostats.rand . . . . .                     | 32 |
| resultstantoRfunc . . . . .                       | 33 |

|                                   |           |
|-----------------------------------|-----------|
| resultstantoRfunc.rand . . . . .  | 34        |
| Save.resulttoRDatafile . . . . .  | 35        |
| simulatetrial . . . . .           | 36        |
| Sperarmfunc . . . . .             | 38        |
| stan.logisticmodeltrans . . . . . | 39        |
| Stopboundinf . . . . .            | 40        |
| testing_and_armdropping . . . . . | 41        |
| Timetrend.fun . . . . .           | 43        |
| Trial.simulation . . . . .        | 44        |
| trtbias . . . . .                 | 45        |
| trteffect . . . . .               | 45        |
| varfunc . . . . .                 | 46        |
| <b>Index</b>                      | <b>47</b> |

---

BayesianPlatformDesignTimeTrend-package

*The 'BayesianPlatformDesignTimeTrend' package.*

---

## Description

This package simulates the multi-arm multi-stage or platform trial with bayesian approach using the 'rstan' package, which provides the R interface for to the stan. The package uses Thall's and Trippa's randomisation approach for Bayesian adaptive randomisation. In addition, the time trend problem of platform trial can be studied in this package. There is a demo for multi-arm multi-stage trial for two different null scenario in this package.

## References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. <https://mc-stan.org>

---

AdaptiveRandomisation *AdaptiveRandomisation*

---

## Description

This is a function doing the randomisation process. This Function generates the Sequence for patient allocation to each arm, patient outcomes.

**Usage**

```

AdaptiveRandomisation(
  Fixratio,
  rand.algo,
  K,
  n.new,
  randomprob,
  treatmentindex,
  groupwise.response.probs,
  group,
  armleft,
  max.deviation,
  trend_add_or_multip,
  trend.function,
  trend.effect,
  ns,
  Fixratiocontrol
)

```

**Arguments**

|                          |   |
|--------------------------|---|
| Fixratio                 | A indicator TRUE/FALSE  |
| rand.algo                | Randomisation algorithm: "Coin": Biased coin; "Urn": Urn method   |
| K                        | Total number of arms at the beginning   |
| n.new                    | The cohort size   |
| randomprob               | A named vector of randomisation probability to each arm   |
| treatmentindex           | The vector of treatment arm index excluding the control arm whose index is 0  |
| groupwise.response.probs | A matrix of response probability of each arm  |
| group                    | The current stage   |
| armleft                  | The number of treatment left in the platform (>2)   |
| max.deviation            | Tuning parameter of using urn randomisation method.   |
| trend_add_or_multip      | How time trend affects the true response probability: "add" or "mult"   |
| trend.function           | The function returns time trend effect regarding to different time trend pattern  |
| trend.effect             | The strength of time trend effect as a parameter in trend.function()  |
| ns                       | A vector of accumulated number of patient at each stage   |
| Fixratiocontrol          | A numeric value indicating the weight of control in randomisation. Eg. 1 means equal randomisation, 2 means thw number of patients allocated to control is twice as large as other treatment arm. |

**Value**

A list of patient allocation and patient outcome nstage: A vector of the number of patients allocated to each arm ystage: A vector of the patients outcome after treating with each arm znew: A vector of treatment index assigned to each patient in the current cohort ynew: A vector of outcome index record for each patient after treatment in the current cohort

**Author(s)**

Ziyan Wang

**References**

Mass weighted urn design—a new randomization algorithm for unequal allocations. Zhao, Wenle. Contemporary clinical trials 43 (2015): 209-216.

**Examples**

```
AdaptiveRandomisation(
  Fixratio = FALSE,
  rand.algo = "Urn",
  K = 2,
  n.new = 30,
  randomprob = matrix(c(0.5, 0.5), ncol = 2, dimnames = list(c(),c("1","2"))),
  treatmentindex = 1,
  groupwise.response.probs = matrix(rep(c(0.4, 0.4), 5), byrow = TRUE, ncol = 2, nrow = 5),
  group = 1,
  armleft = 2,
  max.deviation = 3,
  trend_add_or_multip = "mult",
  trend.function = function(ns, group, i, trend.effect) {delta = 0; return(delta)},
  trend.effect = c(0, 0),
  ns = c(30, 60, 90, 120, 150),
  Fixratiocontrol = NA)

AdaptiveRandomisation(
  Fixratio = TRUE,
  rand.algo = "Urn",
  K = 4,
  n.new = 30,
  randomprob = NA,
  treatmentindex = c(1,3),
  groupwise.response.probs = matrix(rep(c(0.4, 0.4,0.4, 0.4), 5), byrow = TRUE, ncol = 4, nrow = 5),
  group = 1,
  armleft = 3,
  max.deviation = 3,
  trend_add_or_multip = "mult",
  trend.function = function(ns, group, i, trend.effect) {delta = 0; return(delta)},
  trend.effect = c(0, 0),
  ns = c(30, 60, 90, 120, 150),
  Fixratiocontrol = 1)
```

alphaspending

*alphaspending*

---

**Description**

This function estimates the mean error rate spent at each interim analysis for a trial Example usage:

1. `sapply(res = result, fun = alphaspending)` will generate list of the proportion of trial replicates are stopped at each stage for all scenarios in result where result is a list containing output data for different scenario
2. `sapply(sapply(result,FUN = alphaspending),sum)` will generate the type I error rate or power for all scenario on the result list
3. `alpha(result)` generate the proportion of trial replicates are stopped at each stage where result is the output data for one specific scenario
4. `sum(alpha(result))` will generate the type I error rate or power for a specific scenario

**Usage**

```
alphaspending(res)
```

**Arguments**

`res`                    A list of output matrix of a number of trial replicates

**Value**

The error rate at each interim analysis

**Author(s)**

Ziyan Wang

**Examples**

```
## Not run: alphaspending(res)
```

ARmethod

*ARmethod***Description**

This function adjusts the posterior randomisation probability for each arm using many approaches. Currently Thall's approach and Trippa's approach are used. Double biased coin and other method will be added in the next version.

**Usage**

```
ARmethod(
  BARmethod,
  group,
  stats,
  post.prob.btcontrol,
  K,
  n,
  tuningparameter = NA,
  c = NA,
  a = NA,
  b = NA,
  post.prob.best,
  max.ar = NA,
  armleft,
  treatmentindex
)
```

**Arguments**

|                     |  |
|---------------------|--|
| BARmethod           | The indicator of which adaptive randomisation method is used                         |
| group               | The current stage  |
| stats               | The output matrix  |
| post.prob.btcontrol | The vector of posterior probability of each active treatment arm better than control |
| K                   | Total number of arms at the beginning  |
| n                   | The vector of sample size for each arm   |
| tuningparameter     | The tuning parameter indicator for Thall's approach                                  |
| c                   | The tuning parameter for Thall's approach  |
| a                   | The hyperparameter parameter for Trippa's approach                                   |
| b                   | The hyperparameter parameter for Trippa's approach                                   |
| post.prob.best      | Posterior probability of each arm to be the best                                     |

|                |   |
|----------------|---|
| max.ar         | The upper boundary for randomisation ratio for each arm, which is used in Thall's approach since Trippa's approach has protection on control arm. |
| armleft        | The number of treatment left in the platform (>2)   |
| treatmentindex | The vector of treatment arm index excluding the control arm whose index is 0  |

**Value**

randomprob: The vector of adjusted randomisation probability to each arm

**Author(s)**

Ziyan Wang

**References**

Bayesian adaptive randomized trial design for patients with recurrent glioblastoma. Trippa, Lorenzo, Eudocia Q. Lee, Patrick Y. Wen, Tracy T. Batchelor, Timothy Cloughesy, Giovanni Parmigiani, and Brian M. Alexander. *Journal of Clinical Oncology* 30, no. 26 (2012): 3258. A simulation study of outcome adaptive randomization in multi-arm clinical trials. Wathen, J. Kyle, and Peter F. Thall. *Clinical Trials* 14, no. 5 (2017): 432-440.

**Examples**

```
ARmethod(
  BARmethod = "Thall",
  group = 1,
  stats = matrix(rep(NA, 40), ncol = 8, nrow = 5),
  post.prob.btcontrol = 0.5,
  K = 2,
  n = c(30, 30),
  tuningparameter = "fixed",
  c = 1,
  post.prob.best = c(0.5, 0.5),
  max.ar = 0.75,
  armleft = 2,
  treatmentindex = 1)
```

```
ARmethod(
  BARmethod = "Trippa",
  group = 1,
  stats = matrix(rep(NA, 40), ncol = 8, nrow = 5),
  post.prob.btcontrol = c(0.5, 0.6),
  K = 3,
  n = c(30, 30, 40),
  tuningparameter = NA,
  c = NA,
  a = 3,
  b = 0.75,
  post.prob.best = c(0.3, 0.3, 0.4),
  max.ar = NA,
  armleft = 3,
```



```
treatmentindex = c(1, 2))
```

---

Boundaryconstruction *Boundaryconstruction*

---

### Description

This function constructs the stopping boundary based on input information

### Usage

```
Boundaryconstruction(Stopbound.inf = Stopbound.inf, ns = ns)
```

### Arguments

`Stopbound.inf` The list of stop boundary information for more see [Stopboundinf](#)  
`ns` A vector of accumulated number of patient at each stage

### Value

A list of the futility boundary and the efficacy boundary

### Author(s)

Ziyan Wang

### Examples

```
Stopbound.inf=list(Stop.type="Early-Pocock",Boundary.type="Symmetric",cutoff=c(0.9928,0.0072))
ns=c(60,120,180,240,300)
Boundaryconstruction(Stopbound.inf, ns)
```

---

conjunctivepower\_or\_FWER  
*conjunctivepower\_or\_FWER*

---

### Description

This function reads in the output matrix of a number of trial replicates to calculate the Family wise error rate or Conjunctive power

### Usage

```
conjunctivepower_or_FWER(res, scenario, test.type)
```

**Arguments**

|           |   |
|-----------|---|
| res       | A list of output matrix of a number of trial replicates   |
| scenario  | The true scenario used to generate the res list   |
| test.type | The indicator of whether using one side or two side testing. Please make sure that the input test.type does not conflicts to the data. Otherwise the conjunctive power calculation is wrong |

**Value**

Family wise error rate or Conjunctive power

**Examples**

```
## Not run: conjunctivepower_or_FWER(res)
```

---

dataloginformd

*Cutoff screening example: the details of grid*

---

**Description**

Details of grid including family wise error rate of a cutoff value, the cutoff value and the square of cutoff value for modelling and prediction

**Usage**

```
dataloginformd
```

**Format**

A data frame with 24 rows and 3 variables:

tpIE FWER

cutoff Cutoff value

cutoff2 Square of cutoff value

---

 demo\_Cutoffscreening *demo\_Cutoffscreening*


---

## Description

This function does a cutoff screening for trial simulation.

## Usage

```
demo_Cutoffscreening(
  ntrials = 1000,
  trial.fun = simulatetrial,
  grid.inf = list(start = c(0.9, 0.95, 1), extendlength = 15),
  input.info = list(response.probs = c(0.4, 0.4), ns = c(30, 60, 90, 120, 150), max.ar =
    0.75, rand.algo = "Urn", max.deviation = 3, test.type = "Twoside", model.inf =
    list(model = "tlr", ibb.inf = list(pi.star = 0.5, pess = 2, betabinomialmodel =
      ibetabinomial.post), tlr.inf = list(beta0_prior_mu = 0, beta1_prior_mu = 0,
      beta0_prior_sigma = 2.5, beta1_prior_sigma = 2.5, beta0_df = 7, beta1_df = 7, reg.inf
      = "main", variable.inf = "Fixeffect")), Stop.type = "Early-Pocock", Boundary.type =
      "Symmetric", Random.inf = list(Fixratio = FALSE,
      Fixratiocontrol = NA,
      BARMethod = "Thall", Thall.tuning.inf = list(tuningparameter = "Fixed", fixvalue =
      1)), trend.inf = list(trend.type = "step", trend.effect = c(0, 0),
      trend_add_or_multip = "mult")),
  cl = 2
)
```

## Arguments

|                         |   |
|-------------------------|---|
| <code>ntrials</code>    | A numeric variable indicating how many trial replicates you want to run               |
| <code>trial.fun</code>  | The function of trial simulation, related to <code>MainFunction.R</code>              |
| <code>grid.inf</code>   | A list of grid information to create start grid and extend grid for cutoff screening. |
| <code>input.info</code> | A list of input information including all information required for trial simulation.  |
| <code>cl</code>         | A numeric variable indicating how many cores you want to use in parallel programming. |

## Value

A vector of recommended cutoff. The final value is the latest recommended value. A plot for all tested cutoff and error rate

## Author(s)

Ziyang Wang

**Examples**

```
demo_Cutoffscreening(ntrials = 2, cl = 2,
  grid.inf = list(start = c(0.9, 0.95, 1), extendlength = 2))
```

---

demo\_Cutoffscreening.GP

*A demo for cutoff screening using Bayesian optimisation*

---

**Description**

This function does a cutoff screening for trial simulation using Bayesian optimisation.

**Usage**

```
demo_Cutoffscreening.GP(
  ntrials = 1000,
  trial.fun = simulatetrial,
  grid.inf = list(start.length = 10, grid.min = NULL, grid.max = NULL, confidence.level =
    0.95, grid.length = 5000, change.scale = FALSE, noise = TRUE, errorrate = 0.1,
    simulationerror = 0.01, iter.max = 15, plotornot = FALSE),
  power.type = NA,
  response.probs.alt = NA,
  input.info = list(response.probs.null = c(0.4, 0.4, 0.4, 0.4), ns = c(120, 240, 360,
    480, 600), max.ar = 0.85, rand.algo = "Urn", max.deviation = 3, test.type =
    "Twoside", model.inf = list(model = "tlr", ibb.inf = list(pi.star = 0.5, pess = 2,
    betabinomialmodel = ibetabinomial.post), tlr.inf = list(beta0_prior_mu = 0,
    beta1_prior_mu = 0, beta0_prior_sigma = 2.5, beta1_prior_sigma = 2.5, beta0_df = 7,
    beta1_df = 7, reg.inf = "main", variable.inf = "Fixeffect")), Stop.type =
    "Early-OBF", Boundary.type = "Symmetric",
    Random.inf = list(Fixratio = FALSE,
    Fixratiocontrol = NA, BARMethod = "Thall", Thall.tuning.inf = list(tuningparameter =
    "Unfixed", fixvalue = 1)), trend.inf = list(trend.type = "step", trend.effect = c(0,
    0, 0, 0), trend_add_or_multip = "mult")),
  cl = 2
)
```

**Arguments**

|           |  |
|-----------|--|
| ntrials   | A numeric variable indicating how many trial replicates you want to run  |
| trial.fun | The function of trial simulation, related to MainFunction.R  |
| grid.inf  | A list of grid information to create start grid and extend grid for cutoff screening. 'start.length' is the size of start grid. Default is 10. 'grid.min' A numeric value or vector (for asymmetric boundary) indicating the lower bound of the grid for screening. For asymmetric boundary, the first value is efficacy minimum value and the second value is futility minimum value. 'grid.max' A numeric value or vector (for asymmetric boundary) indicating the upper bound of the grid for |

screening. For asymmetric boundary, the first value is efficacy maximum value and the second value is futility maximum value. 'errorrate' refers to the target of type I error rate or family-wise error rate. Default is 0.05. User can change it to 0.1 for FWER if they think 0.05 is too conservative. The per-hypothesis type I error equals errorrate / (K-1) where (K-1) is the number of treatment arms. 'confidence.level' is a numeric value indicating the confidence level of estimate. Default is 0.95. 'grid.length' A numeric value indicating the grid resolution. Default is 5000 for symmetric boundary. For asymmetric boundary, the length of grid is 101 for both efficacy grid and futility grid. A numeric value indicating the grid resolution. Default is 5000 for symmetric boundary. For asymmetric boundary, the length of grid is 101 for both efficacy grid and futility grid. 'change.scale' is a logic value indicating whether we want to change scale when doing Gaussian process. Default is FALSE. 'noise' is a logic value indicating whether the input x is noisy. Default is TRUE. 'simulationerror' is a numeric value indicating the tolerable error for simulated type I error rate. Default is 0.01, 'iter.max' is a numeric value indicating the maximum number of evaluations. Default is 15. 'plotornot' is a logic value indicating whether the errorrate vs grid plot needed to be generated at each iteration. Default is FALSE.

|                    |   |
|--------------------|---|
| power.type         | A indicator of which type of power we need to optimise when tuning the cutoff value for asymmetric boundary. Default is NA (Symmetric boundary). The choice of power type is Conjunctive power ("Conjunctive") and Disconjunctive power ("Disconjunctive"). In a two arm trial design, these power type are the same. |
| response.probs.alt | A vector of response probability of each arm under the alternative scenario. This is used for power optimisation when tuning the cutoff values for asymmetric boundary. Default is NA.  |
| input.info         | A list of input information including all information required for trial simulation.  |
| c1                 | A numeric variable indicating how many cores you want to use in parallel programming.   |

### Value

A vector of recommended cutoff. The final value is the latest recommended value. A plot for all tested cutoff and error rate

### Author(s)

Ziyan Wang

### Examples

```
#Two arm asymmetric boundary screening. Default is OBF boundary.
demo_Cutoffscreening.GP(ntrials = 2, c1 = 2,
  power.type = NA,
  response.probs.alt = NA,
  grid.inf = list(
    start.length = 10,
```

```

confidence.level = 0.95,
grid.length = 5000,
change.scale = FALSE,
noise = TRUE,
errorrate = 0.1,
simulationerror = 0.01,
iter.max = 15,
plotornot = FALSE))

#Four arm asymmetric OBF boundary screening where conjunctive power is optimised.
demo_Cutoffscreening.GP(ntrials = 2, cl = 2,
power.type = "Conjunctive",
response.probs.alt = c(0.4,0.6,0.6,0.4),
grid.inf = list(
start.length = 10,
confidence.level = 0.95,
grid.length = 101,
change.scale = FALSE,
noise = TRUE,
errorrate = 0.1,
simulationerror = 0.01,
iter.max = 15,
plotornot = FALSE))
input.info = list(
response.probs.null = c(0.4,0.4,0.4,0.4),
ns = c(120, 240, 360, 480, 600),
max.ar = 0.85,
rand.algo = "Urn",
max.deviation = 3,
test.type = "Twoside",
model.inf = list(
model = "tlr",
ibb.inf = list(
pi.star = 0.5,
pess = 2,
betabinomialmodel = ibetabinomial.post
),
tlr.inf = list(
beta0_prior_mu = 0,
beta1_prior_mu = 0,
beta0_prior_sigma = 2.5,
beta1_prior_sigma = 2.5,
beta0_df = 7,
beta1_df = 7,
reg.inf = "main",
variable.inf = "Fixeffect"
)
),
Stop.type = "Early-OBF",
Boundary.type = "Asymmetric",
Random.inf = list(
Fixratio = FALSE,
Fixratiocontrol = NA,

```

```
BARmethod = "Thall",
Thall.tuning.inf = list(tuningparameter = "Unfixed", fixvalue = 1)
),
trend.inf = list(
trend.type = "step",
trend.effect = c(0, 0, 0, 0),
trend_add_or_multip = "mult"
)
)
```

---

demo\_multscenario      *demo\_multscenario*

---

### Description

This is a demo function simulating multi-arm multi-stage design with two different null scenarios where the response probability of control is 0.15 and 0.4, respectively. The clinically meaningful increment on probability scale is 0.2. The stopping boundary is the OBF. The cutoff vector in the demo is tuned to keep Type I error rate to be 0.05. The output data can be saved as .RData file

### Usage

```
demo_multscenario(ntrials = 1000, cl = 2, save_data = FALSE)
```

### Arguments

|           |   |
|-----------|---|
| ntrials   | A numeric value. The number of total trial replicates for each scenario.              |
| cl        | A numeric variable indicating how many cores you want to use in parallel programming. |
| save_data | An indicator of whether the output data need to be saved. Default is FALSE.           |

### Value

A list of data for plotting. One is results of trial replicates for all scenarios. The other one is a data frame containing all summarised evaluation metrics for all scenarios

### Author(s)

Ziyan Wang

### Examples

```
demo_multscenario(ntrials = 2, cl = 2, save_data = FALSE)
```

---

```
disconjunctivepowerfunc
      disconjunctivepowerfunc
```

---

**Description**

This function reads in the output matrix of a number of trial replicates to calculate the Family wise error rate or disconjunctive power

**Usage**

```
disconjunctivepowerfunc(res)
```

**Arguments**

```
res          A list of output matrix of a number of trial replicates
```

**Value**

Disconjunctive power

**Examples**

```
## Not run: disconjunctivepowerfunc(res)
```

---

```
GP.optim          GP.optim: optimiser to give the next cutoff for evaluation
```

---

**Description**

A function to predict the next cutoff value for evaluation.

**Usage**

```
GP.optim(
  x,
  y.t1E,
  y.pow = NA,
  ESS = NA,
  errorrate = 0.05,
  confidence.level = 0.95,
  grid.length = 1000,
  change.scale = FALSE,
  noise = T,
  grid.min,
  grid.max,
  Boundary.type = "Symmetric"
)
```



**Arguments**

|                  |  |
|------------------|--|
| x                | A numeric vector of cutoff data  |
| y.t1E            | A numeric vector of type I error rate data   |
| y.pow            | A numeric vector of power data. You can input conjunctive, disjunctive and marginal power data. Default is NA. Only used when Boundary.type == "Asymmetric"  |
| ESS              | A matrix of effective sample size. This is only called for asymmetric boundary cutoff screening. Default is NA for symmetric boundary. The first column is the ESS for different cutoff pair under the null scenario, the second column is the ESS for different cutoff pair under the alternative scenario. |
| errorrate        | 'errorrate' refers to the target of type I error rate or family-wise error rate. Default is 0.05. User can change it to 0.1 for FWER if they think 0.05 is too conservative. The per-hypothesis type I error equals errorrate / (K-1) where (K-1) is the number of treatment arms.                           |
| confidence.level | A numeric value indicating the confidence level of estimate. Default is 0.95   |
| grid.length      | A numeric value indicating the grid resolution. Default is 1000.   |
| change.scale     | A logic value indicating whether we want to change scale when doing Gaussian process. Default is FALSE.  |
| noise            | A logic value indicating whether the input x is noisy. Default is TRUE.  |
| grid.min         | A numeric value or vector (for asymmetric boundary) indicating the lower bound of the grid for screening. For asymmetric boundary, the first value is efficacy minimum value and the second value is futility minimum value.   |
| grid.max         | A numeric value or vector (for asymmetric boundary) indicating the upper bound of the grid for screening. For asymmetric boundary, the first value is efficacy maximum value and the second value is futility maximum value.   |
| Boundary.type    | A text indicating what type of boundary used. Default is "Symmetric"   |

**Value**

A list including the next cutoff value for evaluation `next.cutoff` and a list of predictions for screening grid.

**Author(s)**

Ziyan Wang

**References**

Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. CRC press. Gramacy, R.B., 2020. Bayesian optimization for adaptive experimental design: A review. IEEE access, 8, 13937-13948. Greenhill, S., Rana, S., Gupta, S., Vellanki, P., & Venkatesh, S. (2020).

**Examples**

```
x = c(7.123968, 6.449631, 1.984406,
3.507463, 4.972510, 2.925768,
5.816682, 4.367796,
7.349160, 1.113648)
y.t1E = c(0.0396, 0.0450,
0.5116, 0.2172,
0.1040, 0.3058,
0.0592, 0.1384,
0.0296, 0.7936)
grid.min=1
grid.max=8
GP.res=GP.optim(x=x, y.t1E=y.t1E, errorrate = 0.1, grid.min = grid.min, grid.max = grid.max)
GP.res$next.cutoff
```

```
x = data.frame(matrix(c(
0.9563408, 0.006295626,
0.9669739, 0.014395030,
0.9959410, 0.034858339,
0.9635357, 0.048435579,
0.9794314, 0.021659226,
0.9552201, 0.018442535,
0.9655374, 0.035281833,
0.9837123, 0.010656442,
0.9974910, 0.047741842,
0.9989172, 0.012982826), byrow=TRUE, ncol = 2))
y.t1E = c(0.3044, 0.2938, 0.2573, 0.4780, 0.2923, 0.3733, 0.4263, 0.1962, 0.2941, 0.1131)
y.pow = c(0.8300, 0.8239, 0.7102, 0.7291, 0.8205, 0.7984, 0.7709, 0.8418, 0.6359, 0.5609)
ESS = data.frame(matrix(c(
594.672, 560.580,
596.148, 566.328,
597.840, 590.124,
590.052, 544.800,
597.024, 574.716,
593.952, 554.580,
593.676, 554.400,
598.500, 583.896,
595.740, 590.520,
599.580, 598.644), byrow=TRUE, ncol=2))
grid.min=c(0.95,0)
grid.max=c(1,0.05)
GP.res_asy=GP.optim(x=x, y.t1E=y.t1E, y.pow=y.pow, ESS=ESS,errorrate = 0.1,
grid.min = grid.min, grid.max = grid.max, Boundary.type="Asymmetric")
GP.res_asy$next.cutoff
```

**Description**

This function calculates the posterior probability of each active treatment arm better than control using betabinomial model

**Usage**

```
ibetabinomial.post(n, y, pi.star = 0.5, pess = 2)
```

**Arguments**

|         |  |
|---------|--|
| n       | A vector of treated patients for each arm (The first element is for control)         |
| y       | A vector of treated patient outcomes for each arm (The first element is for control) |
| pi.star | The prior response probability. The default is 0.5                                   |
| pess    | The effective sample size of beta prior. The default is 2                            |

**Value**

A vector posterior probability of each active treatment arm better than control

**Author(s)**

Ziyan Wang

**Examples**

```
n <- c(20,20,20,20)
y <- c(12,12,12,6)
ibetabinomial.post(n, y, pi.star = 0.5, pess = 2)
#[1] 0.5000000 0.5000000 0.0308018
```

---

Initializetrialparameter

*Initializetrialparameter*

---

**Description**

This function initialises the inner parameter used in simulate.trial function

**Usage**

```
Initializetrialparameter(response.probs, ns)
```

**Arguments**

|                |   |
|----------------|---|
| response.probs | A vector of response probability of each arm            |
| ns             | A vector of accumulated number of patient at each stage |

**Value**

A list of initialised parameters including the number of arms for this trial 'K', the number of arm active 'armleft', the index of treatment arm 'treatmentindex', the vector of total number of patients allocated to each arm 'n' the number of total number of patients survived for each arm 'y1', the matrix for true response probability of each arm at each stage 'groupwise.response.probs' which is required for the time trend study, the vector of randomisation probability for each arm 'randomprob', the array of arm assignment for each patient 'z', the array of outcome for each patient 'y', the array of the stage index for each patient 'group\_indicator', the matrix of the probability of each arm to be the best at each stage 'post.prob.best.mat'.

**Author(s)**

Ziyan Wang

**Examples**

```
Initializetrialparameter(response.probs = c(0.4,0.6), ns = c(15,30,45,60,75,90))

#$K
#[1] 2

#$armleft
#[1] 2

#$treatmentindex
#[1] 1

#$n
#[1] 0 0

#$y1
#[1] 0 0

#$groupwise.response.probs
#      [,1] [,2]
#[1,] 0.4 0.6
#[2,] 0.4 0.6
#[3,] 0.4 0.6
#[4,] 0.4 0.6
#[5,] 0.4 0.6
#[6,] 0.4 0.6

#$randomprob
#      1 2
#[1,] 0.5 0.5

#$z
#numeric(0)

#$y
#numeric(0)
```

```

#$group_indicator
#numeric(0)

#$post.prob.best.mat
# 0 1
#[1,] 0 0
#[2,] 0 0
#[3,] 0 0
#[4,] 0 0
#[5,] 0 0
#[6,] 0 0

```

---

|         |                |
|---------|----------------|
| intbias | <i>intbias</i> |
|---------|----------------|

---

**Description**

This function estimates the mean bias of treatment - stage interaction effect

**Usage**

```
intbias(res)
```

**Arguments**

res                    A list of output matrix of a number of trial replicates

**Value**

A matrix of mean treatment - stage interaction effect bias

**Examples**

```
## Not run: intbias(res)
```

---

|          |                 |
|----------|-----------------|
| Meanfunc | <i>Meanfunc</i> |
|----------|-----------------|

---

**Description**

This function reads in the output matrix of a number of trial replicates to calculate mean treatment effect estimate.

**Usage**

```
Meanfunc(res)
```

**Arguments**

res                    A list of output matrix of a number of trial replicates

**Value**

Mean treatment effect estimates of each treatment arm

**Examples**

```
## Not run: Meanfunc(res)
```

---

modelinf.fun

*modelinf.fun*

---

**Description**

This function summarize the input parameters describing the model for analysis and transfer them into a list

**Usage**

```
modelinf.fun(
  model = "tlr",
  ibb.inf = list(pi.star = 0.5, pess = 2, betabinomialmodel = ibetabinomial.post),
  tlr.inf = list(beta0_prior_mu = 0, beta1_prior_mu = 0, beta0_prior_sigma = 2.5,
    beta1_prior_sigma = 2.5, beta0_df = 7, beta1_df = 7, reg.inf = "main", variable.inf =
    "Fixeffect")
)
```

**Arguments**

model                    The statistical model. ibb: betabinomial model / tlr: logistic model

ibb.inf                    The list of information for betabinomial model including: betabinomialmodel: The betabinomial model, pi.star: prior response rate, pess: prior effective sample size

tlr.inf                    The list of information for logistic model including: The mean (mu), variance (sigma), degree of freedom (df) of the intercept and the main effect of the linear terms in logistic model. reg.inf: The type of linear function in logistic model. variable.inf: Fixeffect/Mixeffect. Indicating whether a mix effect model is used (for time trend effect modelling)

**Value**

A list of model information including model, ibb.inf and tlr.inf

**Author(s)**

Ziyan Wang

**Examples**

```
modelinf.fun(model = "tlr",
  ibb.inf = list(pi.star = 0.5,
    pess = 2,
    betabinomialmodel = ibetabinomial.post),
  tlr.inf = list(beta0_prior_mu = 0,
    beta1_prior_mu = 0,
    beta0_prior_sigma = 2.5,
    beta1_prior_sigma = 2.5,
    beta0_df = 7,
    beta1_df = 7,
    reg.inf = "main",
    variable.inf = "Fixeffect"
  ))
```

---

Nfunc

*Nfunc*

---

**Description**

This function reads in the output matrix of a number of trial replicates to calculate mean estimate of total number of patients allocated to each arm

**Usage**

```
Nfunc(res)
```

**Arguments**

res                    A list of output matrix of a number of trial replicates

**Value**

The mean estimate of total number of patients allocated to each arm

**Examples**

```
## Not run: Nfunc(res)
```

---

 OPC\_alt

---

*Operation characteristic table for alternative scenario*


---

### Description

Operation characteristic table for alternative scenario using main + continuousstage model. The time trend pattern is step. The strength of time trend is 0.1 equally for all arm. The effect of time trend on true response probability is multiplicative.

### Usage

OPC\_alt

### Format

A data frame with 3 rows and 16 variables:

Type.I.Error.or.Power Power

Bias.1 Treatment effect bias for treatment 1

Bias.2 Treatment effect bias for treatment 2

Bias.3 Treatment effect bias for treatment 3

rMSE.1 Rooted mean squared error for treatment 1

rMSE.2 Rooted mean squared error for treatment 2

rMSE.3 Rooted mean squared error for treatment 3

N.per.arm.1 Mean total number of patient allocated to control

N.per.arm.2 Mean total number of patient allocated to treatment 1

N.per.arm.3 Mean total number of patient allocated to treatment 2

N.per.arm.4 Mean total number of patient allocated to treatment 3

Survive.per.arm.1 Mean total number of patient allocated to control

Survive.per.arm.2 Mean total number of patient survived when using treatment 1

Survive.per.arm.3 Mean total number of patient survived when using treatment 2

Survive.per.arm.4 Mean total number of patient survived when using treatment 3

N Mean total number of patient in a trial



---

 OPC\_null

---

*Operation characteristic table for null scenario*


---

### Description

Operation characteristic table for null scenario using main and main + continuousstage model. The main effect model was run for a null scenario with and without time trend. The time trend pattern is step. The strength of time trend is 0.1 equally for all arm. The effect of time trend on true response probability is multiplicative.

### Usage

OPC\_null

### Format

A data frame with 3 rows and 16 variables:

Type.I.Error.or.Power Family wise error rate

Bias.1 Treatment effect bias for treatment 1

Bias.2 Treatment effect bias for treatment 2

Bias.3 Treatment effect bias for treatment 3

rMSE.1 Rooted mean squared error for treatment 1

rMSE.2 Rooted mean squared error for treatment 2

rMSE.3 Rooted mean squared error for treatment 3

N.per.arm.1 Mean total number of patient allocated to control

N.per.arm.2 Mean total number of patient allocated to treatment 1

N.per.arm.3 Mean total number of patient allocated to treatment 2

N.per.arm.4 Mean total number of patient allocated to treatment 3

Survive.per.arm.1 Mean total number of patient allocated to control

Survive.per.arm.2 Mean total number of patient survived when using treatment 1

Survive.per.arm.3 Mean total number of patient survived when using treatment 2

Survive.per.arm.4 Mean total number of patient survived when using treatment 3

N Mean total number of patient in a trial

---

OPC\_Trial.simulation    *Operation characteristic table for Trial.simulation() null scenario*

---

### Description

Operation characteristic table for null scenario using main model.

### Usage

OPC\_Trial.simulation

### Format

A data frame with 1 rows and 8 variables:

Type.I.Error.or.Power    Power

Bias    Treatment effect bias for treatment 1

rMSE    Rooted mean squared error for treatment effect 1

N.per.arm.1    Mean total number of patient allocated to control

N.per.arm.2    Mean total number of patient allocated to treatment 1

Survive.per.arm.1    Mean total number of patient allocated to control

Survive.per.arm.2    Mean total number of patient survived when using treatment 1

N    Mean total number of patient in a trial

---

optimdata\_asy    *A list of data from Gaussian process and trial simulation for asymmetric cutoff screening.*

---

### Description

A list of data from Gaussian process and trial simulation for asymmetric cutoff screening.

### Usage

optimdata\_asy

**Format**

A list with four element:

`next.cutoff` The cutoff value for the next evaluation

`prediction` A list of values from Gaussian process model

`ESS` A two column twenty five rows matrix with the effective sample size for each cutoff pair under both null (first column) and alternative (second column) scenario

`testedata` A data frame containing each tested cutoff pair (column two and three for efficacy and futility, respectively), their FWER under null (column one) and conjunctive power under alternative (column four)

---

|                            |   |
|----------------------------|---|
| <code>optimdata_sym</code> | <i>A list of data from Gaussian process for symmetric cutoff screening.</i> |
|----------------------------|---|

---

**Description**

A list of data from Gaussian process for symmetric cutoff screening.

**Usage**

```
optimdata_sym
```

**Format**

A list with two element:

`next.cutoff` The cutoff value for the next evaluation

`prediction` A list of values from Gaussian process model

`tpIE` A vector of type I error rate data

`cutoff` A vector of cutoff data

---

|                                       |                                 |
|---------------------------------------|---------------------------------|
| <code>OutputStats.initialising</code> | <i>OutputStats.initialising</i> |
|---------------------------------------|---------------------------------|

---

**Description**

This function initializes the output matrix including all evaluation metrics based on input information

**Usage**

```
OutputStats.initialising(variable.inf, reg.inf, ns, K)
```

**Arguments**

|                           |  |
|---------------------------|--|
| <code>variable.inf</code> | The parameter information in the model   |
| <code>reg.inf</code>      | The model information. For the fixed effect model, the input of <code>reg.inf</code> can be <code>main</code> , <code>main + stage_continuous</code> , <code>main * stage_continuous</code> , <code>main + stage_discrete</code> , <code>main * stage_discrete</code> . For the mixed effect model, the <code>reg.inf</code> is invalid. |
| <code>ns</code>           | A vector of accumulated number of patient at each stage  |
| <code>K</code>            | Total number of arm including control  |

**Value**

The empty output matrix including different evaluation metrics.

**Author(s)**

Ziyan Wang

**Examples**

```
OutputStats.initialising(
  variable.inf = "Fixeffect",
  reg.inf = "main",
  ns = c(15, 30, 45, 60, 75),
  K = 2)
```

---

`perHtypeIerror_marginalpowerfunc`  
*perHtypeIerror\_powerfunc*

---

**Description**

This function reads in the output matrix of a number of trial replicates to calculate the error rate

**Usage**

```
perHtypeIerror_marginalpowerfunc(res)
```

**Arguments**

|                  |   |
|------------------|---|
| <code>res</code> | A list of output matrix of a number of trial replicates |
|------------------|---|

**Value**

"per-hypothesis" Type I error rate or power (marginal power) for each treatment - control comparison

**Author(s)**

Ziyan Wang

**Examples**

```
## Not run: perHtypeIError_powerfunc(res)
```

---

```
predictedtpIEinformd Cutoff screening example: the predicted value from quadratic model
```

---

**Description**

The predicted value from quadratic model for family wise error rate vs cutoff value plotting

**Usage**

```
predictedtpIEinformd
```

**Format**

A data frame with 1001 rows and 1 variables:

```
predictedtpIEinformd The predicted FWER value of a large grid
```

---

```
Randomisation.inf Randomisation.inf
```

---

**Description**

This function checks the validity of the randomisation information input

**Usage**

```
Randomisation.inf(
  Random.inf = list(Fixratio = FALSE, Fixratiocontrol = NA, BARmethod = "Thall",
    Thall.tuning.inf = list(tuningparameter = "Fixed", fixvalue = 1), Trippa.tuning.inf =
      list(a = 10, b = 0.75))
)
```

**Arguments**

`Random.inf` A list of adaptive randomisation information. `'Fixratio'` a indicator of whether the randomisation process uses fix ratio. Default is FALSE. `'Fixratiocontrol'` the numerical value indicating the randomisation weight of the control arm compared to the treatment arms. Default is NA for `Fixratio = FALSE`. `'BARmethod'` the Bayesian adaptive randomisation type. Default is "Thall" indicating the use of Thall's approach in the randomisation process. The other value is 'Trippa'. `'Thall.tuning.inf'` the list of tuning parameter for Thall's approach including `'tuningparameter'` (Default is "Fixed" indicating that the tuning parameter is fixed for all stages) and `'fixvalue'` (Default is 1).

**Value**

A list of input randomisation information

**Author(s)**

Ziyan Wang

**Examples**

```
Randomisation.inf(Random.inf = list(
  Fixratio = FALSE,
  Fixratiocontrol = NA,
  BARmethod = "Thall",
  Thall.tuning.inf = list(tuningparameter = "Fixed", fixvalue = 1),
  Trippa.tuning.inf = list(a = NA, b = NA)
))
```

---

|                     |  |
|---------------------|--|
| recommandloginformd | <i>Cutoff screening example: the recommended grid value at each time point</i> |
|---------------------|--|

---

**Description**

he recommended grid value at each time point. There are 20 cutoff value explored

**Usage**

```
recommandloginformd
```

**Format**

A data frame with 20 rows and 1 variables:

```
recommandloginformd The cutoff value at each time point
```

---

|                |                       |
|----------------|-----------------------|
| resultrtostats | <i>resultrtostats</i> |
|----------------|-----------------------|

---

**Description**

This is an inner function of the function [resultstantoRfunc](#)

**Usage**

```
resultrstats(  
  trteff = NA,  
  treatmentindex = NA,  
  armleft,  
  K,  
  group,  
  reg.inf,  
  fit,  
  ns  
)
```

**Arguments**

|                |  |
|----------------|--|
| trteff         | Stan posterior samples of treatment effect sample distribution   |
| treatmentindex | A vector of treatment index at the beginning of a trial          |
| armleft        | The number of treatment left in the platform (>2)                |
| K              | Total number of arms at the beginning                            |
| group          | The current stage  |
| reg.inf        | The information of how much accumulated information will be used |
| fit            | The stan output  |
| ns             | A vector of accumulated number of patient at each stage          |

**Value**

A list of stan result inference stats1: A vector of posterior probability for all treatment arms including dropped and active treatment arm stats4: The mean treatment effect estimate of each treatment compared to control stats5: The variance of treatment effect estimate of each treatment compared to control post.prob.btcontrol: a vector including Posterior probability of each active treatment arm better than control

**Author(s)**

Ziyan Wang

**Examples**

```
## Not run: resultrstats(trteff = NA, treatmentindex = NA, armleft, K, group, reg.inf, fit, ns)
```

---

resultrtostats.rand    *resultrtostats.rand*

---

### Description

The inner function of function [resultstantoRfunc.rand](#)

### Usage

```
resultrtostats.rand(
  trteff = NA,
  treatmentindex = NA,
  armleft,
  K,
  group,
  fit,
  ns
)
```

### Arguments

|                |  |
|----------------|--|
| trteff         | Stan posterior samples of treatment effect sample distribution |
| treatmentindex | A vector of treatment index at the beginning of a trial        |
| armleft        | The number of treatment left in the platform (>2)              |
| K              | Total number of arms at the beginning                          |
| group          | The current stage  |
| fit            | The stan output  |
| ns             | A vector of accumulated number of patient at each stage        |

### Value

A list of stan result inference stats1: A vector of posterior probability for all treatment arms including dropped and active treatment arm stats4: The mean treatment effect estimate of each treatment compared to control stats5: The variance of treatment effect estimate of each treatment compared to control post.prob.btcontrol: a vector including Posterior probability of each active treatment arm better than control

### Author(s)

Ziyan Wang

### Examples

```
## Not run: resultrtostats.rand(trteff = NA, treatmentindex = NA, armleft, K, group, fit, ns)
```



---

|                  |                         |
|------------------|-------------------------|
| resultstanoRfunc | <i>resultstanoRfunc</i> |
|------------------|-------------------------|

---

## Description

This function summarise the fix effect stan output data to and transform them to be readable.

## Usage

```
resultstanoRfunc(
  group,
  reg.inf,
  variable.inf,
  fit,
  armleft,
  treatmentindex,
  K,
  ns
)
```

## Arguments

|                |  |
|----------------|--|
| group          | The current stage  |
| reg.inf        | The information of how much accumulated information will be used           |
| variable.inf   | The information of whether to use random effect model or fix effect model. |
| fit            | The stan output  |
| armleft        | The number of treatment left in the platform (>2)                          |
| treatmentindex | A vector of treatment index at the beginning of a trial                    |
| K              | Total number of arms at the beginning                                      |
| ns             | A vector of accumulated number of patient at each stage                    |

## Value

A list of stan result inference stats1: A vector of posterior probability for all treatment arms including dropped and active treatment arm stats4: The mean treatment effect estimate of each treatment compared to control stats5: The variance of treatment effect estimate of each treatment compared to control post.prob.btcontrol: a vector including Posterior probability of each active treatment arm better than control stats6: A vector of mean stage (time trend) effect estimate stats7: A vector of mean treatment - stage (time trend) interaction effect estimate sampefftotal: The posterior samples of response probability of each active arm on logit scale. This can be transformed to probit scale by using `inv.logit()` function.

## Author(s)

Ziyang Wang

**Examples**

```
## Not run: resultstantoRfunc(group, reg.inf, fit, armleft, treatmentindex, K, ns)
```

---

```
resultstantoRfunc.rand
      resultstantoRfunc.rand
```

---

**Description**

This function summarise the mix effect stan output data to and transform them to be readable.

**Usage**

```
resultstantoRfunc.rand(group, fit, armleft, treatmentindex, K, ns)
```

**Arguments**

|                |   |
|----------------|---|
| group          | The current stage                                       |
| fit            | The stan output   |
| armleft        | The number of treatment left in the platform (>2)       |
| treatmentindex | A vector of treatment index at the beginning of a trial |
| K              | Total number of arms at the beginning                   |
| ns             | A vector of accumulated number of patient at each stage |

**Value**

A list of stan result inference stats1: A vector of posterior probability for all treatment arms including dropped and active treatment arm stats4: The mean treatment effect estimate of each treatment compared to control stats5: The variance of treatment effect estimate of each treatment compared to control post.prob.btcontrol: a vector including Posterior probability of each active treatment arm better than control stats6: A vector of mean stage (time trend) effect estimate stats7: A vector of mean treatment - stage (time trend) interaction effect estimate sampefftotal: The posterior samples of response probability of each active arm on logit scale. This can be transformed to probit scale by using `inv.logit()` function.

**Author(s)**

Ziyan Wang

**Examples**

```
## Not run: resultstantoRfunc.rand(group, fit, armleft, treatmentindex, K, ns)
```

---

 Save.resulttoRDatafile

*Save.resulttoRDatafile*


---

## Description

This function generates the name of file for output table and dataset

## Usage

```
Save.resulttoRDatafile(
  input.info = list(response.probs = c(0.4, 0.4), ns = c(30, 60, 90, 120, 150), max.ar =
    0.75, rand.type = "Urn", max.deviation = 3, model.inf = list(model = "tlr", ibb.inf =
    list(pi.star = 0.5, pess = 2, betabinomialmodel = ibetabinomial.post), tlr.inf =
    list(beta0_prior_mu = 0, beta1_prior_mu = 0, beta0_prior_sigma = 2.5,
    beta1_prior_sigma = 2.5, beta0_df = 7, beta1_df = 7, reg.inf = "main", variable.inf =
    "Fixeffect")), Stopbound.inf = Stopboundinf(Stop.type = "Early-Pocock", Boundary.type
    = "Symmetric", cutoff = c(0.99,
    0.01)), Random.inf = list(Fixratio = FALSE,
    Fixratiocontrol = NA, BARmethod = "Thall", Thall.tuning.inf = list(tuningparameter =
    "Fixed", fixvalue = 1)), trend.inf = list(trend.type = "step", trend.effect = c(0,
    0), trend_add_or_multip = "mult"))
)
```

## Arguments

`input.info` A list of input information require for trial simulation

## Value

A list of name for table and dataset

## Author(s)

Ziyan Wang

## Examples

```
Save.resulttoRDatafile(
  input.info = list(
    response.probs = c(0.4, 0.4),
    ns = c(30, 60, 90, 120, 150),
    max.ar = 0.75,
    rand.type = "Urn",
    max.deviation = 3,
    model.inf = list(
      model = "tlr",
      ibb.inf = list(
```

```

    pi.star = 0.5,
    pess = 2,
    betabinomialmodel = ibetabinomial.post
  ),
  tlr.inf = list(
    beta0_prior_mu = 0,
    beta1_prior_mu = 0,
    beta0_prior_sigma = 2.5,
    beta1_prior_sigma = 2.5,
    beta0_df = 7,
    beta1_df = 7,
    reg.inf = "main",
    variable.inf = "Fixeffect"
  )
),
Stop.type = "Early-Pocock",
Boundary.type = "Symmetric",
Random.inf = list(
  Fixratio = FALSE,
  Fixratiocontrol = NA,
  BARmethod = "Thall",
  Thall.tuning.inf = list(tuningparameter = "Fixed", fixvalue = 1)
),
trend.inf = list(
  trend.type = "step",
  trend.effect = c(0, 0),
  trend_add_or_multip = "mult"
)
))

```

---

 simulatetrial

*simulatetrial*


---

## Description

This function simulates a MAMS trial applying adaptive methods where the time trend effect can be studied.

## Usage

```

simulatetrial(
  ii,
  response.probs = c(0.4, 0.4),
  ns = c(30, 60, 90, 120, 150),
  test.type = "Twoside",
  max.ar = 0.75,
  rand.algo = "Urn",
  max.deviation = 3,
  model.inf = list(model = "tlr", ibb.inf = list(pi.star = 0.5, pess = 2,

```

```

betabinomialmodel = ibetabinomial.post), tlr.inf = list(beta0_prior_mu = 0,
beta1_prior_mu = 0, beta0_prior_sigma = 2.5, beta1_prior_sigma = 2.5, beta0_df = 7,
beta1_df = 7, reg.inf = "main", variable.inf = "Fixeffect")),
Stopbound.inf = Stopbound.inf,
Random.inf = Random.inf,
trend.inf = trend.inf
)

```

## Arguments

|                             |   |
|-----------------------------|---|
| <code>ii</code>             | Meaning less parameter but required for foreach function in doParallel package  |
| <code>response.probs</code> | A vector of true response probability for each arm. Default <code>response.probs = c(0.4, 0.4)</code> .   |
| <code>ns</code>             | A vector of accumulated number of patient at each stage. Default is <code>ns = c(30, 60, 90, 120, 150)</code> .   |
| <code>test.type</code>      | A indicator of whether to use one side test or two side test for each treatment-control comparison.   |
| <code>max.ar</code>         | The upper boundary for randomisation ratio for each arm. Default is 0.75 for a two arm trial. The minimum value depends on K where $1 - \text{max.ar} \leq 1/K$ |
| <code>rand.algo</code>      | The method of applying patient allocation with a given randomisation probability vector. Default is "Urn".  |
| <code>max.deviation</code>  | The tuning parameter for Urn randomisation method. Default is 3.  |
| <code>model.inf</code>      | The list of interim data analysis model information for more see <a href="#">modelinf.fun</a>   |
| <code>Stopbound.inf</code>  | The list of stop boundary information for more see <a href="#">Stopboundinf</a>   |
| <code>Random.inf</code>     | The list of Adaptive randomisation information for more see <a href="#">Randomisation.inf</a>   |
| <code>trend.inf</code>      | The list of time trend information  |

## Value

A matrix including all evaluation metrics

## Author(s)

Ziyan Wang

## Examples

```

set.seed(1)
simulatetrial(response.probs = c(0.4, 0.4),
              ns = c(30, 60, 90, 120, 150),
              max.ar = 0.75,
              test.type = "Twoside",
              rand.algo = "Urn",
              max.deviation = 3,
              model.inf = list(
                model = "tlr",
                ibb.inf = list(

```

```

pi.star = 0.5,
pess = 2,
betabinomialmodel = ibetabinomial.post
),
tlr.inf = list(
beta0_prior_mu = 0,
beta1_prior_mu = 0,
beta0_prior_sigma = 2.5,
beta1_prior_sigma = 2.5,
beta0_df = 7,
beta1_df = 7,
reg.inf = "main",
variable.inf = "Fixeffect"
)
),
Stopbound.inf = Stopboundinf(
Stop.type = "Early-Pocock",
Boundary.type = "Symmetric",
cutoff = c(0.99,0.01)
),
Random.inf = list(
Fixratio = FALSE,
Fixratiocontrol = NA,
BARmethod = "Thall",
Thall.tuning.inf = list(tuningparameter = "Fixed", fixvalue = 1),
Trippa.tuning.inf = list(a = 10, b = 0.75)
),
trend.inf = list(
trend.type = "step",
trend.effect = c(0, 0),
trend_add_or_multip = "mult"
))

```

---

Sperarmfunc

*Sperarmfunc*


---

### Description

This function reads in the output matrix of a number of trial replicates to calculate mean total number of survived patients of each arm

### Usage

```
Sperarmfunc(res)
```

### Arguments

res                    A list of output matrix of a number of trial replicates

**Value**

The mean total number of survived patients of each arm

**Examples**

```
## Not run: Sperarmfunc(res)
```

---

```
stan.logisticmodeltrans
      stan.logisticmodeltrans
```

---

**Description**

This function transform the data in trial simulation to the data required for stan modelling

**Usage**

```
stan.logisticmodeltrans(
  z,
  y,
  randomprob,
  group_indicator,
  armleft,
  group,
  variable.inf,
  reg.inf
)
```

**Arguments**

|                              |  |
|------------------------------|--|
| <code>z</code>               | A vector of all treatment index data from the beginning of a trial |
| <code>y</code>               | A vector of all outcome data from the beginning of a trial         |
| <code>randomprob</code>      | A named vector of randomisation probability to each arm            |
| <code>group_indicator</code> | A vector for the stage at which each patient was treated           |
| <code>armleft</code>         | The number of treatment left in the platform (>2)                  |
| <code>group</code>           | The current stage  |
| <code>variable.inf</code>    | Fixeffect/Mixeffect for logistic model parameter                   |
| <code>reg.inf</code>         | The information of how much accumulated information will be used   |

**Value**

A list of information require for the stan model including: `zdropped`: The vector of treatment index for each patient whose treatment arm is active at current stage. `ydropped`: The vector of outcome index for each patient whose treatment arm is active at current stage. `Ndropped`: The total number of patients that are treated with active treatment arms at current stage. `group_indicator_dropped`: The vector of stage index for each patient whose treatment arm is active at current stage. `zlevel`: The active treatment arm index at current stage `xdummy`: A design matrix transformed from `zdropped` and `group_indicator_dropped` for modelling

**Author(s)**

Ziyan Wang

**Examples**

```
stan.logisticmodeltrans(
  z = c(1,2,1,2,2,1,2,1),
  y = c(0,0,0,0,1,1,1,1),
  randomprob = matrix(c( 0.5, 0.5), ncol = 2, dimnames = list(c("Stage1"), c("1", "2"))),
  group_indicator = c(1,1,1,1,1,1,1,1),
  armleft = 2,
  group = 1,
  variable.inf = "Fixeffect",
  reg.inf = "main")
```

---

Stopboundinf

*Stopboundinf*

---

**Description**

This function summaries and checks stopping boundary information.

**Usage**

```
Stopboundinf(
  Stop.type = "Early-Pocock",
  Boundary.type = "Symmetric",
  cutoff = c(0.9928, 0.0072)
)
```

**Arguments**

`Stop.type` The type of stopping boundary should be "Early-Pocock", "Early-Obf" and "Noearly". Default is "Early-Pocock" which is the Pocock boundary with early stopping.



`Boundary.type` Whether the futility boundary and the efficacy boundary are the same conservative. Default is "Symmetric" which means they are as conservative as each other. `Boundary.type = "Asymmetric"` means that the efficacy boundary and the futility boundary are not as conservative as each other

`cutoff` = `c(cutoff1, cutoff2)`. A numerical vector of cutoff value for each boundary. The first element is the efficacy boundary cutoff. The second element is the futility boundary cutoff  $\Pr(\theta_{1} > \theta_{0|D_n}) > \text{cutoff1}$ . Should input the cutoff1 for efficacy boundary as the first element  $\Pr(\theta_{1} < \theta_{0|D_n}) < \text{cutoff2}$ . Should input the cutoff2 for futility boundary as the first element

**Value**

The list of information required for boundary construction function 'Stopbound.inf'

**Author(s)**

Ziyan Wang

**Examples**

```
Stop.type = "Early-Pocock" #(Pocock boundarty is a flat boundary across time)
Boundary.type = "Symmetric"
cutoff = c(0.9928, 0.0072)

Stopbound.inf = Stopboundinf(Stop.type, Boundary.type, cutoff)
#Stopbound.inf
#$Stop.type
# [1] "Early-Pocock"
#$Boundary.type
#[1] "Symmetric"
#$cutoff
# [1] 0.9928 0.0072
```

---

testing\_and\_armdropping

*testing\_and\_armdropping*

---

**Description**

This function makes a decision on whether any active arm should be dropped based on posterior probability and return the vector of decision on each arm, the vector of active arms index and the number of arms left for further study.

**Usage**

```
testing_and_armdropping(
  K,
  armleft,
  post.prob.btcontrol,
  group,
  cutofffeff,
  cutofffful,
  treatmentindex,
  test.type
)
```

**Arguments**

|                     |  |
|---------------------|--|
| K                   | A numeric value indicating the total number of arm at the beginning of trial including both control and treatment.   |
| armleft             | A numeric vector indicating the number of active arms before this interim analysis;  |
| post.prob.btcontrol | A numeric vector of posterior probability of each treatment arm better than control  |
| group               | A numeric value. The current stage index.  |
| cutofffeff          | A numeric vector of the cutoff value at each stage for efficacy boundary.  |
| cutofffful          | A numeric vector of the cutoff value at each stage for futility boundary.  |
| treatmentindex      | A numeric vector of the current active treatment arm index   |
| test.type           | A character indicating which hypothesis testing we are use. "Oneside": $H_0: \pi_k \leq \pi_0$ ; $H_0: \pi_k > \pi_0$ "Twoside": $H_0: \pi_k \leq \pi_0$ ; $H_0: \pi_k \neq \pi_0$ |

**Value**

A list of information including armleft: the number of active arms after this interim analysis; treatmentindex: the index vector of active arm after this interim analysis; stats3: the vector of conclusion on whether null hypothesis is rejected

**Examples**

```
testing_and_armdropping(
  K = 4,
  armleft = 4,
  post.prob.btcontrol = c(0.5,0.99,0.02),
  group = 3,
  cutofffeff = c(1, 0.99, 0.975, 0.96, 0.95),
  cutofffful = c(0, 0.01, 0.025, 0.04, 0.05),
  treatmentindex = c(1,2,3),
  test.type = "Oneside")
```

```

testing_and_armdropping(
  K = 4,
  armleft = 4,
  post.prob.btcontrol = c(0.5,0.99,0.02),
  group = 3,
  cutofffeff = c(1, 0.99, 0.975, 0.96, 0.95),
  cutoffful = c(0, 0.01, 0.025, 0.04, 0.05),
  treatmentindex = c(1,2,3),
  test.type = "Twoside")

```

---

Timetrend.fun

*Timetrend.fun*


---

### Description

This function generate the time trend function based on trend information. This function also check the validity of the input time trend information.

### Usage

```
Timetrend.fun(trend.inf)
```

### Arguments

`trend.inf` The list of information for time trend effect including 'trend.type', 'trend.effect', 'trend\_add\_or\_multip'. 'trend.type' is the shape of time trend. Default is "step". Other types are "linear", "inverse.U.linear", "plateau". "trend.effect" the vector of the strength of time trend for each arm. The first element is for the control arm. "trend\_add\_or\_multip" the pattern of time trend affecting the true response probability. Default is "mult".

### Value

A list containing the time trend function according to input trend.type variable, and a indicator of whether there is a time trend in data generation based on input trend information

### Author(s)

Ziyan Wang

### Examples

```

Timetrend.fun(trend.inf = list(
  trend.type = "step",
  trend.effect = c(0, 0),
  trend_add_or_multip = "mult"
))

```

---

|                  |                         |
|------------------|-------------------------|
| Trial.simulation | <i>Trial simulation</i> |
|------------------|-------------------------|

---

### Description

This function simulates and does final analysis of a trial with one scenario. The time cost of this function depend on the cpu cores of the user's computer.

### Usage

```
Trial.simulation(
  ntrials = 5000,
  trial.fun = simulatetrial,
  input.info = list(response.probs = c(0.4, 0.4), ns = c(30, 60, 90, 120, 150), max.ar =
    0.75, test.type = "Twoside", rand.algo = "Urn", max.deviation = 3, model.inf =
    list(model = "tlr", ibb.inf = list(pi.star = 0.5, pess = 2, betabinomialmodel =
    ibetabinomial.post), tlr.inf = list(beta0_prior_mu = 0, beta1_prior_mu = 0,
    beta0_prior_sigma = 2.5, beta1_prior_sigma = 2.5, beta0_df = 7, beta1_df = 7, reg.inf =
    "main", variable.inf = "Fixeffect")), Stopbound.inf = Stopboundinf(Stop.type =
    "Early-Pocock", Boundary.type = "Symmetric",
    cutoff = c(0.99, 0.01)),
  Random.inf = list(Fixratio = TRUE, Fixratiocontrol = 1, BARMethod = NA,
  Thall.tuning.inf = NA, Trippa.tuning.inf = NA), trend.inf = list(trend.type = "step",
  trend.effect = c(0, 0), trend_add_or_multip = "mult")),
  cl = 2
)
```

### Arguments

|            |   |
|------------|---|
| ntrials    | A numeric variable indicating how many trial replicates you want to run. Default is 5000. |
| trial.fun  | The function of trial simulation for more see <a href="#">simulatetrial</a>               |
| input.info | A list of input information including all information required for trial simulation.      |
| cl         | A numeric variable indicating how many cores you want to use in parallel programming.     |

### Value

A list of output including the final output of each trial replicates called 'result' The analysis result table of the specific trial called 'OPC' and the file name for saving these output on the computer

### Author(s)

Ziyan Wang

**Examples**

```
set.seed(1)
Trial.simulation(ntrials = 2, cl = 2)
```

---

|         |                |
|---------|----------------|
| trtbias | <i>trtbias</i> |
|---------|----------------|

---

**Description**

This function estimates the mean bias of treatment effect

**Usage**

```
trtbias(res, trueeffect)
```

**Arguments**

|            |   |
|------------|---|
| res        | A list of output matrix of a number of trial replicates |
| trueeffect | A vector of true treatment effect in each scenario      |

**Value**

A matrix of mean treatment effect bias

**Examples**

```
## Not run: trtbias(res, trueeffect)
```

---

|           |                  |
|-----------|------------------|
| trteffect | <i>trteffect</i> |
|-----------|------------------|

---

**Description**

This function estimates the mean treatment effect bias and its rooted mean squared error

**Usage**

```
trteffect(res, trueeff)
```

**Arguments**

|         |   |
|---------|---|
| res     | A list of output matrix of a number of trial replicates |
| trueeff | A vector of true treatment effect in each scenario      |

**Value**

A vector of mean treatment effect bias and its rooted mean squared error

**Examples**

```
## Not run: trteffect(res, trueeff)
```

---

varfunc

*varfunc*

---

**Description**

This function reads in the output matrix of a number of trial replicates to calculate the variance of treatment effect estimate.

**Usage**

```
varfunc(res)
```

**Arguments**

res                    A list of output matrix of a number of trial replicates

**Value**

The variance of Treatment effect estimates of each treatment arm

**Examples**

```
## Not run: varfunc(res)
```

# Index

## \* datasets

- dataloginformd, [10](#)
  - OPC\_alt, [24](#)
  - OPC\_null, [25](#)
  - OPC\_Trial.simulation, [26](#)
  - optimdata\_asy, [26](#)
  - optimdata\_sym, [27](#)
  - predictedtpIEinformd, [29](#)
  - recommandloginformd, [30](#)
- AdaptiveRandomisation, [3](#)
- alphaspending, [6](#)
- ARmethod, [7](#)
- BayesianPlatformDesignTimeTrend  
(BayesianPlatformDesignTimeTrend-package),  
[3](#)
- BayesianPlatformDesignTimeTrend-package,  
[3](#)
- Boundaryconstruction, [9](#)
- conjunctivepower\_or\_FWER, [9](#)
- dataloginformd, [10](#)
- demo\_Cutoffscreening, [11](#)
- demo\_Cutoffscreening.GP, [12](#)
- demo\_multscenario, [15](#)
- disconjunctivepowerfunc, [16](#)
- GP.optim, [16](#)
- ibetabinomial.post, [18](#)
- Initializetrialparameter, [19](#)
- intbias, [21](#)
- Meanfunc, [21](#)
- modelinf.fun, [22](#), [37](#)
- Nfunc, [23](#)
- OPC\_alt, [24](#)
- OPC\_null, [25](#)
- OPC\_Trial.simulation, [26](#)
- optimdata\_asy, [26](#)
- optimdata\_sym, [27](#)
- OutputStats.initialising, [27](#)
- perHtypeIError\_marginalpowerfunc, [28](#)
- predictedtpIEinformd, [29](#)
- Randomisation.inf, [29](#), [37](#)
- recommandloginformd, [30](#)
- resultrtostats, [30](#)
- resultrtostats.rand, [32](#)
- resultstantoRfunc, [30](#), [33](#)
- resultstantoRfunc.rand, [32](#), [34](#)
- Save.resulttoRDatafile, [35](#)
- simulatetrial, [36](#), [44](#)
- Sperarmfunc, [38](#)
- stan.logisticmodeltrans, [39](#)
- Stopboundinf, [9](#), [37](#), [40](#)
- testing\_and\_armdropping, [41](#)
- Timetrend.fun, [43](#)
- Trial.simulation, [44](#)
- trtbias, [45](#)
- trteffect, [45](#)
- varfunc, [46](#)