Package ‘ASSISTant’

December 2, 2022

Type Package

Title Adaptive Subgroup Selection in Group Sequential Trials

Version 1.4.3

Date 2022-11-30

VignetteBuilder knitr

URL https://github.com/bnaras/ASSISTant

BugReports https://github.com/bnaras/ASSISTant/issues

Description Clinical trial design for subgroup selection in three-stage group sequential trial as described in Lai, Lavori and Liao (2014, <doi:10.1016/j.cct.2014.09.001>). Includes facilities for design, exploration and analysis of such trials. An implementation of the initial DEFUSE-3 trial is also provided as a vignette.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.2

Imports R6, mvtnorm, knitr, magrittr, dplyr

Suggests rmarkdown

NeedsCompilation no

Author Tze Leung Lai [ctb],
Philip Lavori [aut],
Olivia Liao [aut],
Balasubramanian Narasimhan [aut, cre],
Ka Wai Tsang [aut]

Maintainer Balasubramanian Narasimhan <naras@stat.Stanford.EDU>

Repository CRAN

Date/Publication 2022-12-02 09:30:09 UTC
**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSISTant</td>
<td>2</td>
</tr>
<tr>
<td>ASSISTDesign</td>
<td>3</td>
</tr>
<tr>
<td>ASSISTDesignB</td>
<td>6</td>
</tr>
<tr>
<td>ASSISTDesignC</td>
<td>8</td>
</tr>
<tr>
<td>colNamesForStage</td>
<td>10</td>
</tr>
<tr>
<td>computeMeanAndSD</td>
<td>11</td>
</tr>
<tr>
<td>computeMHPBoundaries</td>
<td>11</td>
</tr>
<tr>
<td>computeMHPBoundaryITT</td>
<td>12</td>
</tr>
<tr>
<td>conformParameters</td>
<td>12</td>
</tr>
<tr>
<td>DEFUSE3Design</td>
<td>13</td>
</tr>
<tr>
<td>generateDiscreteData</td>
<td>16</td>
</tr>
<tr>
<td>generateNormalData</td>
<td>17</td>
</tr>
<tr>
<td>groupSampleSize</td>
<td>18</td>
</tr>
<tr>
<td>LLL.SETTINGS</td>
<td>19</td>
</tr>
<tr>
<td>mHP.b</td>
<td>19</td>
</tr>
<tr>
<td>mHP.btilde</td>
<td>20</td>
</tr>
<tr>
<td>mHP.c</td>
<td>21</td>
</tr>
<tr>
<td>wilcoxon</td>
<td>21</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSISTant</td>
<td>23</td>
</tr>
</tbody>
</table>

**Description**

`ASSISTant` is a package that implements a three-stage adaptive clinical trial design with provision for subgroup selection where the treatment may be effective; see Lai, Lavori and Liao (doi:10.1016/j.cct.2014.09.001). The main design object is an R6 class that can be instantiated and manipulated to obtain the operating characteristics. A vignette is provided showing the use of this package for designing the DEFUSE-3 trial, described in the paper by Lai, Lavori and Liao. The package contains everything necessary to reproduce the results of the paper.

**References**


ASSISTDesign

A class to encapsulate the adaptive clinical trial design of Lai, Lavori and Liao

Description

ASSISTDesign objects are used to design, simulate and analyze adaptive group sequential clinical trial with three stages. For details refer to the paper *Adaptive Choice of Patient Subgroup for Comparing Two Treatments* by Tze Leung Lai and Philip W. Lavori and Olivia Yueh-Wen Liao. Contemporary Clinical Trials, Vol. 39, No. 2, pp 191-200 (2014).

Methods

Public methods:

- ASSISTDesign$new()
- ASSISTDesign$getDesignParameters()
- ASSISTDesign$getTrialParameters()
- ASSISTDesign$getBoundaries()
- ASSISTDesign$setBoundaries()
- ASSISTDesign$print()
- ASSISTDesign$computeCriticalValues()
- ASSISTDesign$explore()
- ASSISTDesign$performInterimLook()
- ASSISTDesign$analyze()
- ASSISTDesign$summary()
- ASSISTDesign$clone()

Method new(): Create a new ASSISTDesign instance using the parameters specified.

Usage:

ASSISTDesign$new(
  designParameters,
  trialParameters,
  discreteData = FALSE,
  boundaries
)

Arguments:

designParameters parameters of the experimental design. Must contain apropriate distributions to sample from, if discreteData = TRUE
trialParameters the trial parameters, such as sample size etc.
discreteData a flag indicating that a discrete distribution is to be used for the Rankin scores
boundaries decision boundaries to use for interim looks, a named vector of btilde, b and c values

Returns: a new AssistDesign object
Method getDesignParameters(): return the designParameters field

Usage:
ASSISTDesign$getDesignParameters()

Method getTrialParameters(): return the trialParameters field

Usage:
ASSISTDesign$getTrialParameters()

Method getBoundaries(): return the boundaries field

Usage:
ASSISTDesign$getBoundaries()

Method setBoundaries(): Set the boundaries field

Usage:
ASSISTDesign$setBoundaries(value)
Arguments:
value a named vector of btilde, b and c values

Method print(): Print details of the design to console

Usage:
ASSISTDesign$print()

Method computeCriticalValues(): Compute the critical boundary values $\tilde{b}$, b and c for futility, efficacy and final efficacy decisions. This is time consuming so cache where possible.

Usage:
ASSISTDesign$computeCriticalValues()
Returns: a named vector of critical values with names btilde, b, and c as in the paper

Method explore(): Explore the design using the specified number of simulations and random number seed and other parameters.

Usage:
ASSISTDesign$explore(
  numberOfSimulations = 5000,
rngSeed = 12345,
trueParameters = self$getDesignParameters(),
recordStats = TRUE,
showProgress = TRUE,
fixedSampleSize = FALSE,
saveRawData = FALSE
)
Arguments:
numberOfSimulations default number of simulations is 5000
rngSeed default seed is 12345
trueParameters the state of nature, by default the value of self$getDesignParameters() as would be the case for a Type I error calculation. If changed, would yield power.
recordStats a boolean flag (default TRUE) to record statistics
showProgress a boolean flag to show progress, default TRUE
fixedSampleSize a boolean flag indicating that patients lost after a futile overall look are not
    made up, default FALSE.
saveRawData a flag (default FALSE) to indicate if raw data has to be saved

Returns: a list of results

Method performInterimLook(): Perform an interim look on trial data
Usage:
ASSISTDesign$performInterimLook(
    trialData,
    stage,
    recordStats = FALSE,
    fixedSampleSize = FALSE
)

Arguments:
trialData trial data frame
stage the trial stage
recordStats a boolean flag to record all statistics
fixedSampleSize a flag to use a fixed sample size to account for loss to follow up

Returns: the trial history

Method analyze(): Analyze the exploration data from trial
Usage:
ASSISTDesign$analyze(trialExploration)

Arguments:
trialExploration the result of a call to explore() to simulate the design

Returns: Return a list of summary quantities

Method summary(): Print the operating characteristics of the design using the analysis data
Usage:
ASSISTDesign$summary(analysis)

Arguments:
analysis the analysis result from the analyze() call

Method clone(): The objects of this class are cloneable with this method.
Usage:
ASSISTDesign$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
LLL::SETTINGS for an explanation of trial parameters
Examples

```r
## Not run:
data(LLL.SETTINGS)
prevalence <- LLL.SETTINGS$prevalences$table1
scenario <- LLL.SETTINGS$scenarios$S0
designParameters <- list(prevalence = prevalence,
                          mean = scenario$mean,
                          sd = scenario$sd)
designA <- ASSISTDesign$new(trialParameters = LLL.SETTINGS$trialParameters,
                          designParameters = designParameters)
print(designA)
result <- designA$explore(showProgress = interactive())
analysis <- designA$analyze(result)
designA$summary(analysis)

## End(Not run)
```

ASSISTDesignB

A fixed sample design to compare against the adaptive clinical trial design

Description

ASSISTDesignB objects are used to design a trial with certain characteristics provided in the object instantiation method. This design differs from ASSISTDesign in only how it computes the critical boundaries, how it performs the interim look, and what quantities are computed in a trial run.

Super class

ASSISTant::ASSISTDesign -> ASSISTDesignB

Methods

Public methods:

- ASSISTDesignB$computeCriticalValues()
- ASSISTDesignB$explore()
- ASSISTDesignB$analyze()
- ASSISTDesignB$summary()
- ASSISTDesignB$clone()

Method computeCriticalValues(): Compute the critical boundary value \( c_\alpha \)

Usage:

ASSISTDesignB$computeCriticalValues()

Returns: a named vector of a single value containing the value for \( c \)

Method explore(): Explore the design using the specified number of simulations, random number seed, and further parameters.
ASSISTDesignB

Usage:
ASSISTDesignB$explore(
  numberOfSimulations = 100,
  rngSeed = 12345,
  trueParameters = self$getDesignParameters(),
  showProgress = TRUE,
  saveRawData = FALSE
)

Arguments:
numberOfSimulations default number of simulations is 100
rngSeed default seed is 12345
trueParameters the state of nature, by default the value of self$getDesignParameters()
as would be the case for a Type I error calculation. If changed, would yield power.
showProgress a boolean flag to show progress, default TRUE
saveRawData a flag (default FALSE) to indicate if raw data has to be saved

Returns: a list of results

Method analyze(): Analyze the exploration data from trial

Usage:
ASSISTDesignB$analyze(trialExploration)

Arguments:
trialExploration the result of a call to explore() to simulate the design

Returns: Return a list of summary quantities

Method summary(): Print the operating characteristics of the design using the analysis data

Usage:
ASSISTDesignB$summary(analysis)

Arguments:
analysis the analysis result from the analyze() call

Method clone(): The objects of this class are cloneable with this method.

Usage:
ASSISTDesignB$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

ASSISTDesign which is a superclass of this object
Examples

```r
## Not run:
data(LLL.SETTINGS)
prevalence <- LLL.SETTINGS$prevalences$table1
scenario <- LLL.SETTINGS$scenarios$S0
designParameters <- list(prevalence = prevalence,
                         mean = scenario$mean,
                         sd = scenario$sd)
designB <- ASSISTDesignB$new(trialParameters = LLL.SETTINGS$trialParameters,
                             designParameters = designParameters)
print(designB)
## A realistic design uses 5000 simulations or more!
result <- designB$explore(showProgress = interactive())
analysis <- designB$analyze(result)
designB$summary(analysis)
```

## End(Not run)

## For full examples, try:
## browseURL(system.file("full_doc/ASSISTant.html", package="ASSISTant"))

ASSISTDesignC

A fixed sample RCT design to compare against the adaptive clinical trial design of Lai, Lavori and Liao.

Description

ASSISTDesignC objects are used to design a trial with certain characteristics provided in the object instantiation method. This design differs from ASSISTDesign in only how it computes the critical boundaries, how it performs the interim look, and what quantities are computed in a trial run.

Super classes

ASSISTant::ASSISTDesignC

Methods

Public methods:

- `ASSISTDesignC$computeCriticalValues()`
- `ASSISTDesignC$explore()`
- `ASSISTDesignC$analyze()`
- `ASSISTDesignC$summary()`
- `ASSISTDesignC$clone()`

Method `computeCriticalValues()`: Compute the critical boundary values \( \hat{b} \), \( b \) and \( c \) for futility, efficacy and final efficacy decisions. This is time consuming so cache where possible.

Usage:
ASSISTDesignC$computeCriticalValues()

Returns: a named list containing the critical value cAlpha

**Method** `explore()`: Explore the design using the specified number of simulations and random number seed and other parameters.

**Usage:**

```r
ASSISTDesignC$explore(
  numberOfSimulations = 5000,
  rngSeed = 12345,
  trueParameters = self$getDesignParameters(),
  showProgress = TRUE,
  saveRawData = FALSE
)
```

**Arguments:**

- `numberOfSimulations` default number of simulations is 5000
- `rngSeed` default seed is 12345
- `trueParameters` the state of nature, by default the value of `self$getDesignParameters()` as would be the case for a Type I error calculation. If changed, would yield power.
- `showProgress` a boolean flag to show progress, default `TRUE`
- `saveRawData` a flag (default `FALSE`) to indicate if raw data has to be saved

**Returns:** a list of results

**Method** `analyze()`: Analyze the design given the `trialExploration` data

**Usage:**

```r
ASSISTDesignC$analyze(trialExploration)
```

**Arguments:**

- `trialExploration` the results from a call to `explore()` to simulate the design

**Returns:** a named list of rejections

**Method** `summary()`: Print the operating characteristics of the design using the analysis data

**Usage:**

```r
ASSISTDesignC$summary(analysis)
```

**Arguments:**

- `analysis` the analysis result from the `analyze()` call

**Returns:** no value, just print

**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**

```r
ASSISTDesignC$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.
colNamesForStage

Return a vector of column names for statistics for a given stage

Description

Return a vector of column names for statistics for a given stage

Usage

colNamesForStage(stage, J)

Arguments

stage the trial stage (1 to 3 inclusive).

J the number of subgroups

Value

a character vector of the column names
**computeMeanAndSD**

Compute the mean and sd of a discrete Rankin distribution

**Description**
Compute the mean and sd of a discrete Rankin distribution

**Usage**
```r
computeMeanAndSD(probVec = rep(1, 7L), support = 0L:6L)
```

**Arguments**
- `probVec`: a probability vector of length equal to length of support, default is uniform
- `support`: a vector of support values (default 0:6 for Rankin Scores)

**Value**
a named vector of mean and sd

**computeMHPBoundaries**
Compute the three modified Haybittle-Peto boundaries

**Description**
Compute the three modified Haybittle-Peto boundaries

**Usage**
```r
computeMHPBoundaries(prevalence, N, alpha, beta, eps, futilityOnly = FALSE)
```

**Arguments**
- `prevalence`: the vector of prevalences between 0 and 1 summing to 1. `J`, the number of groups, is implicitly the length of this vector and should be at least 2.
- `N`: a three-vector of total sample size at each stage
- `alpha`: the type I error
- `beta`: the type II error
- `eps`: the fraction (between 0 and 1) of the type I error to spend in the interim stages 1 and 2
- `futilityOnly`: a logical value indicating only the futility boundary is to be computed; default FALSE

**Value**
a named vector of three values containing \( \tilde{b}, b, c \)
**computeMHPBoundaryITT**  Compute the three modified Haybittle-Peto boundaries and effect size

**Description**

Compute the three modified Haybittle-Peto boundaries and effect size

**Usage**

```r
computeMHPBoundaryITT(prevalence, alpha)
```

**Arguments**

- `prevalence` the vector of prevalences between 0 and 1 summing to 1. $J$, the number of groups, is implicitly the length of this vector and should be at least 2.
- `alpha` the type I error

**Value**

a named vector of a single value containing the value for $c$

**conformParameters**  Conform designParameters so that weights are turned in to probabilities, the null and control distributions are proper matrices etc.

**Description**

Conform designParameters so that weights are turned in to probabilities, the null and control distributions are proper matrices etc.

**Usage**

```r
conformParameters(plist, discreteData = FALSE)
```

**Arguments**

- `plist` the parameter list
- `discreteData` flag if data is discrete

**Value**

the modified parameter list
**The DEFUSE3 design**

**Description**

DEFUSE3Design is a slight variant of the adaptive clinical trial design of Lai, Lavori and Liao. Simulation is used to compute the expected maximum sample size and the boundary for early futility is adjusted to account as well.

**Super class**

`ASSISTant::ASSISTDesign` -> DEFUSE3Design

**Methods**

- **Public methods:**
  - DEFUSE3Design$getOriginalBoundaries()
  - DEFUSE3Design$new()
  - DEFUSE3Design$adjustCriticalValues()
  - DEFUSE3Design$explore()
  - DEFUSE3Design$performInterimLook()
  - DEFUSE3Design$clone()

- **Method getOriginalBoundaries():** Return the original boundaries for the design
  
  **Usage:**
  
  DEFUSE3Design$getOriginalBoundaries()

  **Returns:** a named vector of values for b, btilde and c

- **Method new():** Create a DEFUSE3Design object
  
  **Usage:**
  
  DEFUSE3Design$new(
    designParameters,  
    trialParameters,  
    discreteData = FALSE,  
    numberOfSimulations = 5000,  
    rngSeed = 54321,  
    showProgress = TRUE,  
    trueParameters = NULL,  
    boundaries
  )

  **Arguments:**
  
  designParameters parameters of the experimental design. Must contain appropriate distributions to sample from, if discreteData = TRUE
  
  trialParameters the trial parameters, such as sample size etc.
discreteData a flag indicating that a discrete distribution is to be used for the Rankin scores
numberOfSimulations the number of simulations to use, default 5000
rngSeed the random number generator seed
showProgress a boolean flag to show progress (default TRUE)
trueParameters a list of true parameter values reflecting the state of nature
boundaries decision boundaries to use for interim looks, a named vector of $b_{tilde}$, $b$ and $c$ values

Returns: a new AssistDesign object

Method adjustCriticalValues(): Adjust critical values to account for sample size loss due to futility

Usage:
DEFUSE3Design$adjustCriticalValues(numberOfSimulations, rngSeed, showProgress)

Arguments:
numberOfSimulations the number of simulations to use
rngSeed the random number generator seed
showProgress a boolean flag for showing progress

Returns: the adjusted boundaries

Method explore(): Explore the design using the specified number of simulations and random number seed and other parameters.

Usage:
DEFUSE3Design$explore(
  numberOfSimulations = 5000,
  rngSeed = 12345,
  trueParameters = self$getDesignParameters(),
  recordStats = TRUE,
  showProgress = TRUE,
  saveRawData = FALSE
)

Arguments:
numberOfSimulations default number of simulations is 5000
rngSeed default seed is 12345
trueParameters the state of nature, by default the value of self$getDesignParameters() as would be the case for a Type I error calculation. If changed, would yield power.
recordStats a boolean flag (default TRUE) to record statistics
showProgress a boolean flag to show progress, default TRUE
saveRawData a flag (default FALSE) to indicate if raw data has to be saved

Returns: a list of results

Method performInterimLook(): Perform an interim look for futility

Usage:
DEFUSE3Design$performInterimLook(trialData, stage, recordStats = FALSE)
DEFUSE3Design

**Arguments:**
- trialData trial data frame
- stage the trial stage
- recordStats a boolean flag to record all statistics

**Returns:** the trial history

**Method clone():** The objects of this class are cloneable with this method.

**Usage:**
DEFUSE3Design$clone(deep = FALSE)

**Arguments:**
- deep Whether to make a deep clone.

**See Also**
ASSISTDesign which is a superclass of this object

**Examples**

```r
trialParameters <- list(N = c(200, 340, 476), type1Error = 0.025,
                         eps = 1/2, type2Error = 0.1)
designParameters <- list(nul0 = list(prevalence = rep(1/6, 6), mean = matrix(0, 2, 6),
                           sd = matrix(1, 2, 6)),
                        alt1 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6),
                           c(0.5, 0.4, 0.3, 0, 0, 0)),
                          sd = matrix(1, 2, 6)),
                        alt2 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6),
                           c(0.5, 0.5, 0, 0, 0, 0)),
                          sd = matrix(1, 2, 6)),
                        alt3 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6), rep(0.36, 6)),
                          sd = matrix(1, 2, 6)),
                        alt4 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6), rep(0.30, 6)),
                          sd = matrix(1, 2, 6)),
                       .alt5 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6),
                           c(0.4, 0.3, 0.2, 0, 0, 0)),
                          sd = matrix(1, 2, 6)),
                        alt6 = list(prevalence = rep(1/6, 6), mean = rbind(rep(0, 6),
                           c(0.5, 0.5, 0.3, 0.3, 0.1, 0.1)),
                          sd = matrix(1, 2, 6)))

## Not run:
## A realistic design uses 5000 simulations or more!
defuse3 <- DEFUSE3Design$new(trialParameters = trialParameters,
                           numberOfSimulations = 25,
                           designParameters = designParameters$nul0,
                           showProgress = FALSE)
print(defuse3)
result <- defuse3$explore(showProgress = interactive())
analysis <- defuse3$analyze(result)
print(defuse3$summary(analysis))
```
generateDiscreteData  A data generation function using a discrete distribution for Rankin score rather than a normal distribution

Description

A data generation function using a discrete distribution for Rankin score rather than a normal distribution.

Usage

generateDiscreteData(prevalence, N, support = 0L:6L, ctlDist, trtDist)

Arguments

prevalence a vector of group prevalences (length denoted by J below)
N the sample size to generate
support the support values of the discrete distribution (length K), default 0:6
ctlDist a probability vector of length K denoting the Rankin score distribution for control.
trtDist an K x J probability matrix with each column is the Rankin distribution for the associated group

Value

a three-column data frame of subGroup, trt (0 or 1), and score

Examples

# Simulate data from a discrete distribution for the Rankin scores, # which are typically ordinal integers from 0 to 6 in the following # simulations. So we define a few scenarios.
library(ASSISTant)
null.uniform <- rep(1, 7L) ## uniform on 7 support points
hourglass <- c(1, 2, 2, 1, 2, 2, 1)
inverted.hourglass <- c(2, 1, 1, 2, 1, 2, 1)
bottom.heavy <- c(2, 2, 1, 1, 1, 1, 1)
bottom.heavier <- c(3, 3, 2, 2, 1, 1, 1)
top.heavy <- c(1, 1, 1, 2, 2, 2, 2)
top.heavier <- c(1, 1, 1, 2, 2, 3, 3)
ctlDist <- null.uniform
trtDist <- cbind(null.uniform, null.uniform, hourglass, hourglass) ## 4 groups
generateNormalData

A data generation function along the lines of what was used in the Lai, Lavori, Liao paper. score rather than a normal distribution

description

A data generation function along the lines of what was used in the Lai, Lavori, Liao paper. score rather than a normal distribution

Usage

generateNormalData(prevalence, N, mean, sd)

Arguments

prevalence a vector of group prevalences (length denoted by J below)
N the sample size to generate
mean a 2 x J matrix of means under the null (first row) and alternative for each group
sd a 2 x J matrix of standard deviations under the null (first row) and alternative for each group

Value

a three-column data frame of subGroup, trt (0 or 1), and score
groupSampleSize

Compute the sample size for any group at a stage assuming a nested structure as in the paper.

Description

In the three stage design under consideration, the groups are nested with assumed prevalences and fixed total sample size at each stage. This function returns the sample size for a specified group at a given stage, where the futility stage for the overall group test may be specified along with the chosen subgroup.

Usage

groupSampleSize(
  prevalence,
  N,
  stage,
  group,
  HJFutileAtStage = NA,
  chosenGroup = NA
)

Arguments

prevalence the vector of prevalence, will be normalized if not already so. The length of this vector implicitly indicates the number of groups J.

N an integer vector of length 3 indicating total sample size at each of the three stages

stage the stage of the trial

group the group whose sample size is desired

HJFutileAtStage is the stage at which overall futility occurred. Default NA indicating it did not occur. Also ignored if stage is 1.

chosenGroup the selected group if HJFutilityAtStage is not NA. Ignored if stage is 1.

Value

the sample size for group
**LLL.SETTINGS**

*Design and trial settings used in the Lai, Lavori, Liao paper simulations*

**Description**

A list of design and trial design settings used for analysis and simulations in the Lai, Lavori, Liao paper displayed in Tables 1 and 2. The elements of the list are the following:

- **trialParameters** N the sample size at each of three interim looks, the last being the final one; The length of this also determines the number of interim looks
- **type1Error** the overall type I error
- **eps** the fraction of type I error spent at each interim look
- **type2Error** the type II error desired
- **scenarios** A list of the 10 settings used in the simulations named S0, S1, ..., S10 as in the paper, each with three elements
  - **mean** a $2 \times J$ matrix of means, the first row for the null setting, the second for the alternative
  - **sd** a $2 \times J$ matrix of standard deviations, the first row for the null setting, the second for the alternative
- **prevalences** A list of two elements with prevalence vectors used in the paper; the lengths of these vectors implicitly define the number of groups.
  - **table1** a vector of equal prevalences for six groups used in table 1
  - **table2** a vector of prevalences used in table 2 of the paper

**References**


---

**mHP.b**

*Compute the efficacy boundary (modified Haybittle-Peto) for the first two stages*

**Description**

Compute the efficacy boundary (modified Haybittle-Peto) for the first two stages

**Usage**

```
mHP.b(prevalence, N, cov.J, mu.prime, Sigma.prime, alpha, btilde, theta)
```
Arguments

prevalence  the vector of prevalences between 0 and 1 summing to 1. \( J \), the number of groups, is implicitly the length of this vector and should be at least 2.

\( N \)  a three-vector of total sample size at each stage

cov.\( J \)  the 3 x 3 covariance matrix for \( Z_J \) at each of the three stages

mu.prime  a list of \( J \) mean vectors, each of length \( J - 1 \) representing the conditional means of all the other \( Z_j \) given \( Z_i \). This mean does not account for the conditioned value of \( Z_i \) and so has to be multiplied by that during use!

Sigma.prime  a list of \( J \) covariance matrices, each \( J - 1 \) by \( J - 1 \) representing the conditional covariances all the other \( Z_j \) given \( Z_i \)

alpha  the amount of type I error to spend

btilde  the futility boundary

theta  the effect size on the probability scale

\[
\text{mHP.btilde} \quad \text{Compute the futility boundary (modified Haybittle-Peto) for the first two stages}
\]

Description

The futility boundary \( \tilde{b} \) is computed by solving (under the alternative)

Usage

\[
m\text{HP.btilde}(\text{beta, cov.} J)\]

Arguments

beta  the type II error

cov.\( J \)  the 3 x 3 covariance matrix

Details

\[
P(\bar{Z}_1^J \leq \tilde{b} \text{or } \bar{Z}_2^J \leq \tilde{b}) = \epsilon \beta
\]

where the superscripts denote the stage and \( \epsilon \) is the fraction of the type I error (\( \alpha \)) spent and \( \beta \) is the type II error. We make use of the joint normal density of \( Z_J \) (the overall group) at each of the three stages and the fact that the \( \bar{Z}_j \) is merely a translation of \( Z_j \). So here the calculation is based on a mean of zero and has to be translated during use!
mHP.c

Compute the efficacy boundary (modified Haybittle-Peto) for the final (third) stage

Description

Compute the efficacy boundary (modified Haybittle-Peto) for the final (third) stage

Usage

mHP.c(prevalence, N, cov.J, mu.prime, Sigma.prime, alpha, btilde, b, theta)

Arguments

prevalence the vector of prevalences between 0 and 1 summing to 1. J, the number of groups, is implicitly the length of this vector and should be at least 2.

N a three-vector of total sample size at each stage

cov.J the 3 x 3 covariance matrix for Z, at each of the three stages

mu.prime a list of J mean vectors, each of length J - 1 representing the conditional means of all the other Z given Z. This mean does not account for the conditioned value of Z and so has to be multiplied by that during use!

Sigma.prime a list of J covariance matrices, each J - 1 by J - 1 representing the conditional covariances all the other Z given Z

alpha the amount of type I error to spend

btilde the futility boundary

b the efficacy boundary for the first two stages

theta the effect size on the probability scale

wilcoxon

Compute the standardized Wilcoxon test statistic for two samples

Description

We compute the standardized Wilcoxon test statistic with mean 0 and and standard deviation 1 for samples x and y. The R function stats::wilcox.test() returns the statistic

Usage

wilcoxon(x, y, theta = 0)
Arguments

- x: a sample numeric vector
- y: a sample numeric vector
- theta: a value > 0 but < 1/2.

Details

\[ U = \sum_i R_i - \frac{m(m + 1)}{2} \]

where \( R_i \) are the ranks of the first sample \( x \) of size \( m \). We compute

\[
\frac{(U - mn(1/2 + \theta))}{\sqrt{mn(m + n + 1)/12}}
\]

where \( \theta \) is the alternative hypothesis shift on the probability scale, i.e. \( P(X > Y) = 1/2 + \theta \).

Value

the standardized Wilcoxon statistic
Index

* data
  LLL.SETTINGS, 19

ASSISTant, 2
ASSISTant::ASSISTDesign, 6, 8, 13
ASSISTant::ASSISTDesignB, 8
ASSISTDesign, 3
ASSISTDesignB, 6
ASSISTDesignC, 8
colNamesForStage, 10
computeMeanAndSD, 11
computeMHPBoundaries, 11
computeMHPBoundaryITT, 12
conformParameters, 12
DEFUSE3Design, 13
generateDiscreteData, 16
generateNormalData, 17
groupSampleSize, 18
LLL.SETTINGS, 19

mHP.b, 19
mHP.btilde, 20
mHP.c, 21

stats::wilcox.test(), 21
wilcoxon, 21