Package ‘seqmon’

October 12, 2016

Type Package

Title Group Sequential Design Class for Clinical Trials

Version 2.1

Date 2016-10-06

Author David A Schoenfeld, PhD and Hui Zheng, PhD

Maintainer `Hui Zheng, PhD` <hzheng1@mgh.harvard.edu>

Description S4 class object for creating and managing group sequential designs. It calculates the efficacy and futility boundaries at each look. It allows modifying the design and tracking the design update history.

License MIT + file LICENSE

Depends methods

Suggests

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-12 00:43:08

R topics documented:

  seqmon-package .......................................................... 2
  alphaspend ............................................................... 3
  alphaspendf .............................................................. 3
  betaspend ................................................................. 4
  betaspendf ............................................................... 5
  calcBoundaries .......................................................... 6
  calcBoundaries-methods ................................................. 6
  curtail ................................................................. 7
  curtailDesign ........................................................... 8
  curtailDesign-methods ................................................ 8
  getProbabilities ....................................................... 9
  getProbabilities-methods .............................................. 9
  plotBoundaries ....................................................... 10
Description

a package for creating, monitoring and modifying a group sequential design

Details

The DESCRIPTION file: DESCRIPTION

Author(s)

David A Schoenfeld, PhD and Hui Zheng, PhD

References


alphaspend

Examples

design1<-sequential.design()

alphaspend Function that calculates the upper boundaries for efficacy

Description
Calculates the upper boundaries for efficacy at each look time

Usage
alphaspend(levels, t, int = rep(500, length(t)), tol = 0.005)

Arguments

levels The cumulative alpha spending at each look time
t Normalized look times
int The number of intervals the solution space is partitioned into
tol Tolerance of the solution using uniroot

Value
numeric

Examples

f<- function(t) 0.025*t^4
t<-c(0.33, 0.67, 1)
cum_probs<-f(t)
cum_probs
alphaspend(levels=cum_probs, t, int=rep(500, length(t)), tol=0.005)

alphaspendf The default alpha spending function

Description
The default alpha spending function

Usage
alphaspendf(t)
betaspend

Arguments

t  The normalized look times

Value

numeric

Examples

t<-c(0.33,0.67,1)
alphas<-alphaspendf(t)

## The function is currently defined as
function (t)
0.025 * t^4

betaspend  Function that calculates the lower boundaries for futility

Description

Calculates the lower boundaries for futility at each look

Usage

betaspend(levels, upperboundary, t, int = rep(500, length(t)), noncent, tol = 0.005)

Arguments

levels  The cumulative beta spending at each look time
upperboundary  The upper efficacy boundaries at each look
int  Normalized look times
noncent  The numbers of intervals the solution space is partitioned into
tol  Tolerance of the solution using uniroot

Value

numeric
**Examples**

```r
f <- function(t) 0.025 * t^4
g <- function(t) 0.15 * t^3

# Cumulative betas and alphas

t <- c(0.33, 0.67, 1)
cum_alphas <- f(t)
cum_betas <- g(t)

noncent <- qnorm(0.975) + qnorm(0.85)
upper_boundaries <- alphaspend(cum_alphas, t, int = rep(500, length(t)), tol = 0.005)
lower_boundaries <- betaspend(cum_betas, upper_boundaries, t, int = rep(500, 3), noncent, tol = 0.005)
```

---

**Description**

The default beta spending function

**Usage**

```r
betaspendf(t)
```

**Arguments**

- **t**: The normalized look times

**Value**

numeric

**Examples**

```r
t <- c(0.33, 0.67, 1)
betas <- betaspendf(t)

## The function is currently defined as
function (t) 
0.15 * t^3
```
Function for calculating the efficacy and futility boundaries

Description

Calculates the efficacy and futility boundaries. This only needs to be done once for a new design.

Usage

calcBoundaries(theObject)

Arguments

theObject The sequential design object

Value

numeric

Examples

design1<-sequential.design()
design1<-calcBoundaries(design1)
design1@lowerboundary
design1@upperboundary

Description

Calculates the efficacy and futility boundaries of a new design object. This method should not be run when updating a design.

Methods

This method uses the approach described in Schoenfeld (2001) to calculate the efficacy and futility boundaries.
signature(theObject = "sequential.design")
curtail  

Generic function that calculates the probability to declare efficacy at the end of study given the Z value at the current look

Description

Calculates the probability to declare efficacy at the end of study given the Z value at the current look

Usage

curtail(lower.boundary, upper.boundary, look, t, noncen, current = lower.boundary[look])

Arguments

lower.boundary  lower boundaries
upper.boundary  upper boundaries
look  current look number
t  time of looks
noncen  noncentrality parameter
current  current Z statistic

Value

numeric

Examples

t<-c(0.33, 0.67, 1)
f< - function(t) 0.025*t^4
g<-function(t) 0.20*t^3
a<-f(t)
b<-g(t)
noncen<-pnorm(0.975)+pnorm(0.8)
curtail(b, a, 1, t, noncen)
curtailDesign

Function for calculating the probability for efficacy given known information

description

Calculates the probability for efficacy given the Z value

usage

curtailDesign(theObject, currentZ)

arguments

theObject The sequential design object
currentZ The current Z value

value

numeric

examples

design1 <- sequentialDesign()
design1 <- calcBoundaries(design1)
design1 <- setCurrentLook(design1, 1)
prob1 <- curtailDesign(design1, 1.5)

Description

This method calculates the probability to declare efficacy at the end of the trial at an interim look.

Methods

This method uses the approach described in Schoenfeld (2001) to calculate the probability to declare efficacy at the end of the trial.
getProbabilities

signature(theObject = "sequential.design")

getProbabilities Function that calculates the cumulative probabilities to declare efficacy and futility

Description
Calculates the cumulative probabilities to declare efficacy and futility under the null hypothesis and the alternative hypothesis. It also returns the p-values for declaring efficacy and futility.

Usage
getProbabilities(theObject)

Arguments
theObject The sequential design object

Value
numeric

Examples
design1<-sequential.design()
probs<-getProbabilities(design1)

getProbabilities-methods

~ ~ Methods for Function getProbabilities ~ ~

Description
Calculates the cumulative probabilities to declare efficacy and futility under the null hypothesis and the alternative hypothesis. It also returns the p-values for declaring efficacy and futility.

Methods
This method uses the approach described in Schoenfeld (2001).
plotBoundaries-methods

Signature
signature(theObject = "sequential.design")

Description
Function that plots the efficacy and futility boundaries.

Usage
plotBoundaries(theObject)

Arguments
theObject The sequential design object

Examples
```r
design1 <- sequential.design()
design1 <- calcBoundaries(design1)
plotBoundaries(design1)
```

Description
This method plots the efficacy and futility boundaries.

Methods
signature(theObject = "sequential.design")
printSummary

Function that prints the cumulative probabilities for efficacy and futility

Description

Prints the cumulative probability for efficacy and futility under the null and alternative hypotheses, the corresponding p-values, and the boundaries for Z at each look.

Usage

printSummary(theObject)

Arguments

theObject  The sequential design object

Examples

design1 <- sequential.design()
design1 <- calcBoundaries(design1)
printSummary(design1)

printSummary-methods  ~~ Methods for Function printSummary ~~

Description

This method prints a summary of the design. It prints the efficacy and futility boundaries, the corresponding p-values, and the cumulative alpha and beta spending under the null and alternative hypotheses at each look.

Methods

signature(theObject = "sequential.design")
**seqmon**  
*Generic function that calculates the cumulative alpha and beta spending*

**Description**
Calculates the cumulative alpha and beta spending

**Usage**
```r
seqmon(a, b, t, int = rep(500, length(t)))
```

**Arguments**
- `a`: upper boundaries
- `b`: lower boundaries
- `t`: time of looks
- `int`: number of intervals that the Z-space is partitioned into

**Value**
numeric

**Examples**
```r
t<-c(0.33, 0.67, 1)
f <- function(t) 0.025 * t^4
g <- function(t) 0.20 * t^3
a <- f(t)
b <- g(t)
seqmon(a, b, t, int = rep(500, length(t)))
```

---

**sequential.design**  
The sequential design class

**Description**
The S4 sequential design class

**Usage**
```r
sequential.design(...)```

**Arguments**
...
Details

The sequential design class stores the information of a sequential design, including revision history.

Value

an object of the class "sequential.design"

Author(s)

David A. Schoenfeld, PhD and Hui Zheng, PhD

References


Examples

design1<-sequential.design()


sequential.design-class

Class "sequential.design"

Description

The sequential design class

Objects from the Class

Objects can be created by calls of the form sequential.design(...).

Slots

lower.boundary: Object of class "numeric"
upper.boundary: Object of class "numeric"
times: Object of class "numeric"
noncentrality: Object of class "numeric"
base.alpha.spend: Object of class "function"
base.beta.spend: Object of class "function"
base.alpha.spend.string: Object of class "character"
base.beta.spend.string: Object of class "character"
current.look: Object of class "numeric"
current.alpha.spend: Object of class "numeric"

current.beta.spend: Object of class "numeric"

times.history: Object of class "numeric"

alpha.spent.history: Object of class "numeric"

beta.spent.history: Object of class "numeric"

alpha.func.history: Object of class "numeric"

beta.func.history: Object of class "numeric"

date.stamp: Object of class "POSIXct"

Methods

calcBoundaries signature(theObject = "sequential.design"): ...
curtailDesign signature(theObject = "sequential.design"): ...
getProbabilities signature(theObject = "sequential.design"): ...
plotBoundaries signature(theObject = "sequential.design"): ...
printSummary signature(theObject = "sequential.design"): ...
setAlphaspendfString signature(theObject = "sequential.design"): ...
setBaseAlphaspendf signature(theObject = "sequential.design"): ...
setBaseBetaspendf signature(theObject = "sequential.design"): ...
setBetaspendfString signature(theObject = "sequential.design"): ...
setCurrentLook signature(theObject = "sequential.design"): ...
setDatestamp signature(theObject = "sequential.design"): ...
setNoncentrality signature(theObject = "sequential.design"): ...
setTimes signature(theObject = "sequential.design"): ...
updateDesign signature(theObject = "sequential.design"): ...

Examples

showClass("sequential.design")

---

setAlphaspendfString  Function that Sets the expression of the base alpha spending function as a string

Description

Sets the expression of the base alpha spending function as a string. This function is only used if one needs to display the base alpha spending function as a string. This function DOES NOT update the base alpha spending function. One can use setBaseAlphaspendf() to change the base alpha spending function. The spending functions and their string expressions should be defined only once per object. They should not be updated during any interim update to the design.
setAlphaspendfString-methods

Usage

setAlphaspendfString(theObject, string0)

Arguments

theObject          The sequential design object
string0          The string of the expression of the base alpha spending function. Its argument
                 need to be 't'.

Value

an object of class "sequential.design"

Examples

design1<-sequentialNdesign()
design1<-setAlphaspendfString(design1, '0.025\times t^4')

Description

This method sets the baseline alpha spending function as a string. It is not involved in any of the
calcualtions. It serves only a display purpose.

Methods

signature(theObject = "sequential.design")

setBaseAlphaspendf     Function that sets the base alpha spending function

Description

Sets the base alpha spending function.

Usage

setBaseAlphaspendf(theObject, funct0)
Arguments

theObject  The sequential design object
funct0  The base alpha spending function. It needs to be defined before this method is called.

Value

an object of class "sequential.design"

Examples

design1<-sequentialNdesign()
f1<-function (t) 0.025*t^3.5
design1<-setbasealphaspendf(design1,f1)

Description

This method defines the baseline alpha spending function.

Methods

signature(theObject = "sequential.design")

setBaseAlphaspendf  Function that sets the baseline alpha spending function

Description

Sets the base beta spending function.

Usage

setBaseBetaspndf(theObject, funct0)

Arguments

theObject  The sequential design object
funct0  The base beta spending function. It needs to be defined before this method is called.
Description

This method defines the baseline beta spending function.

Methods

signature(theObject = "sequential.design")

setBetaspendfString  Function that sets the expression of the base beta spending function as a string

Description

Sets the expression of the base beta spending function as a string. This function is only used if one needs to display the base beta spending function as a string. This function DOES NOT update the base beta spending function. One can use setBaseBetaspendf() to change the base beta spending function. The spending functions and their string expressions should be defined only once per object. They should not be updated during any interim update to the design.

Usage

setBetaspendfString(theObject, string)

Arguments

theObject  The sequential design object
string  The string of the expression of the base beta spending function. Its argument need to be 't'.

Value

an object of class "sequential.design"
Examples

```r
design1 <- sequential_design()
design1 <- setBetaspendfString(design1, '0.15*t^3.5')
```

---

**setBetaspendfString-methods**

~~ Methods for Function setBetaspendfString ~~

Description

This method sets the baseline beta spending function as a string. It is not involved in any of the calculations. It serves only a display purpose.

Methods

```r
signature(theObject = "sequential.design")
```

---

**setCurrentLook**

*Function that sets the current look number*

Description

Sets the current look number. The current look is the one that last took place.

Usage

```r
setCurrentLook(theObject, look)
```

Arguments

- `theObject` The sequential design object
- `look` The current look number

Details

The current look is the one that last took place. One can only set the current look forward. If the new current look number attempted is less than the old current look number, no action will take place and the current look number will not be updated.

Value

- an object of class "sequential.design"

Examples

```r
design1 <- sequential_design()
design1 <- setCurrentLook(design1, 2)
```
**setCurrentLook-methods**

~~ Methods for Function setCurrentLook ~~

## Description

This method sets the current look number.

## Methods

The look number can only be set forward. It is used when the design needs to be updated at a certain look.

```r
signature(theObject = "sequential.design")
```

**setDatestamp**

*Function that sets the date stamp of the design object*

## Description

Sets the date stamp of the design object

## Usage

```r
setDatestamp(theObject, date0)
```

## Arguments

- `theObject`: The sequential design object
- `date0`: The date value.

## Value

An object of class "sequential.design"

## Examples

```r
design1 <- sequential.design()
design1 <- setDatestamp(design1, as.POSIXct("2015-10-30"))
```
setNoncentrality

Description
This method sets the time stamp of the creation of the new design. It is not to be used during the course of the trial.

Methods
signature(
  theObject = "sequential.design"
)

setNoncentrality
Function that sets the noncentrality parameter

Description
Sets the noncentrality parameter.

Usage
setNoncentrality(theObject, noncent)

Arguments
theObject The sequential design object
noncent The noncentrality parameter

Details
The noncentrality parameter is the expected drift at the end of the study. For example, if the study has a power of 80% using a one sided Z-test with 2.5% type 1 error, the noncentrality parameter is \( q(0.975) + q(0.8) \), where \( q() \) is the percentile function of the standard normal distribution.

Value
an object of class "sequential.design"

Examples

design1 <- sequential.design()
noncent <- qnorm(0.975, 0, 1) + qnorm(0.8, 0, 1)
design1 <- setNoncentrality(design1, noncent)
Description

This method sets the non-centrality parameter.

Methods

See Schoenfeld (2001) for the definition of the non-centrality parameter.

signature(theObject = "sequential.design")

setTimes Function that sets the look times

Description

Sets the look times. It is to be called only for the initial design, not for updating the design.

Usage

setTimes(theObject, time0)

Arguments

theObject The sequential design object
time0 The look times.

Value

an object of class "sequential.design"

Examples

design1 <- sequential.design()
design1 <- setTimes(design1, c(1,2,3))
setTimes-methods

~ Methods for Function setTimes ~~

Description

This method sets the look times of a new trial. It is not to be used during the course of the trial.

Methods

signature(theObject = "sequential.design")

updateDesign

Function that updates the design

Description

Updates the design. This can be done in the process of the study, when the future look times need to be changed from those originally planned.

Usage

updateDesign(theObject, futureTimes)

Arguments

theObject The sequential design object
futureTimes The future look times.

Details

The efficacy and futility boundaries will be updated according to the new future look times. If the new final look is before the planned final look, the efficacy and futility boundaries will be updated, but the alpha and beta spending functions need not be updated. If the new final look is after the planned final look, the efficacy and futility boundaries will be updated, as well as the alpha and beta spending functions.

Value

an object of class "sequential.design"

Author(s)

David A Schoenfeld, PhD and Hui Zheng, PhD
References


Examples

design1<-sequential.design()
design1<-setTimes(design1,c(1,2))
design1<-calcBoundaries(design1)
design1<-setCurrentLook(design1,1)
design2<-updateDesign(design1,c(3))

Description

This method updates the design at a certain look.

Methods

The trial needs to be updated due to a change in the future look times. The total alpha spending and total beta spending at the end of the trial are not updated. No historical information such as the past look times, the past alpha and beta spent, or the baseline spending function is updated. The alpha or beta spending functions may be updated whenever necessary. The efficacy and futility boundaries are updated when needed. The details are given in Proschan, Lan, and Wittes (2006) and Schoenfeld (2001).
Index

*Topic `textasciitilde` | `textasciitilde`
other possible keyword(s)

*Method

- calcboundaries-methods, 6
- curtailDesign-methods, 8
- getprobabilities-methods, 9
- plotboundaries-methods, 10
- printsummary-methods, 11
- setalphaspendfstring-methods, 15
- setbasealphaspendf-methods, 16
- setbasebetaspendf-methods, 17
- setbetaspendfstring-methods, 18
- setcurrentlook-methods, 19
- setdatestamp-methods, 20
- setnoncentrality-methods, 21
- settimes-methods, 22
- updatedesign-methods, 23
- alphaspend, 3
- alphaspendf, 3
- betaspend, 4
- betaspendf, 5
- calcboundaries, 6
- calcboundaries, sequential.design-method (sequential.design-class), 13
- curtail, 7
- curtaildesign, 8
- curtaildesign, sequential.design-method (sequential.design-class), 13
- printsummary, 11
- seqmon, 12
- sequential.design, 12
- sequential.design, sequential.design-method (sequential.design-class), 13
- sequential.design-class, 13
- setalphaspendfstring, 14
- setalphaspendfstring, sequential.design-method (sequential.design-class), 13
setalphaspendfString-methods, 15
setBaseAlphaspendf, 15
setBaseAlphaspendf, sequential.design-method
  (sequential.design-class), 13
setBaseAlphaspendf-methods, 15
setBaseBetaspndf, 16
setBaseBetaspndf, sequential.design-method
  (sequential.design-class), 13
setBaseBetaspndf-methods, 17
setBetaspndfString, 17
setBetaspndfString, sequential.design-method
  (sequential.design-class), 13
setBetaspndfString-methods, 18
setCurrentLook, 18
setCurrentLook, sequential.design-method
  (sequential.design-class), 13
setCurrentLook-methods, 19
setDatestamp, 19
setDatestamp, sequential.design-method
  (sequential.design-class), 13
setDatestamp-methods, 20
setNoncentrality, 20
setNoncentrality, sequential.design-method
  (sequential.design-class), 13
setNoncentrality-methods, 21
setTimes, 21
setTimes, sequential.design-method
  (sequential.design-class), 13
setTimes-methods, 22
updateDesign, 22
updateDesign, sequential.design-method
  (sequential.design-class), 13
updateDesign-methods, 23