Package 'OneArmPhaseTwoStudy'

November 13, 2017

Type Package

Title Planning, Conducting, and Analysing Single-Arm Phase II Studies

Version 1.0.3

Date 2017-11-13

Author Marius Wirths

Maintainer Johannes Krisam <krisam@imbi.uni-heidelberg.de>

Description Purpose of this package is it to plan, monitor and evaluate
oncological phase II studies. In general this kind of studies are single-arm
trials with planned interim analysis and binary endpoint. To meet the resulting
requirements, the package provides functions to calculate and evaluate 'Simon's
two-stage designs' and 'so-called' 'subset designs'. If you are unfamiliar with
this package a good starting point is to take a closer look at the functions
getSolutions() and getSolutionsSub1().The web-based tool (<https://imbi.shinyapps.io/phaseII-app/>)
extends the functionality of our R package by means of a proper dealing with over- and underrunning.
The R function binom.test of the 'stats' R package and the package 'binom' might be helpful to assess the performance of the corresponding one-stage design as a reference.

License GPL (>= 2)

Depends methods, stats, Rcpp (>= 0.9.11), R (>= 3.1.0)

LinkingTo Rcpp

RcppModules simon, sub1

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-11-13 11:54:38 UTC

R topics documented:

`getCE` *Calculates the conditional error.*

Description

Calculates the conditional error at the interim analysis for a given Simon's design with "k" responses.

Usage

```
getCE(design, k)
```

Arguments

design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums "r1", "n1", "r", "n" and "p0". • r1 = critical value for the first stage (more than "r1" responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than "r" responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis
k	number of responses observed at the interim analysis.

Examples

```
design <- getSolutions()$Solutions[1,]
conditional_error <- getCE(design, 4)
```

---

getCP *Calculates the conditional power.*

Description

Calculates the conditional power for a given Simon's two-stage design in the interim analysis if the number of patients which should be enrolled in the second stage is altert to "n2".

Usage

```
getCP(n2, p1, design, k, mode = 0, alpha = 0.05)
```

Arguments

n2	number of patients to be enrolled in the second stage of the study.
p1	response probability under the alternative hypothesis
design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums "r1", "n1", "r", "n" and "p0". • r1 = critical value for the first stage (more than r1 responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than r responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.
k	number of responses observed at the interim analysis.

mode — a value out of {0,1,2,3} dedicating the methode spending the "rest alpha" (difference between nominal alpha level and actual alpha level for the given design).

- 0 = "rest alpha" is not used.
- 1 = "rest alpha" is spent proportionally.
- 2 = "rest alpha" is spent equally.
- 3 = "rest alpha" is spent only to the worst case scenario (minimal number of responses at the interim analysis so that the study can proceed to the second stage).

alpha — overall significance level the trial was planned for.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

See Also

getN2

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

#Assume 3 responses were observed in the interim analysis.
#Therefore the conditional power is only about 0.55.
#In order to raise the conditional power to 0.8 "n2" has to be increased.

#get the current "n2"
n2 <- design$n - design$n1

#set k to 3 (only 3 responses observed so far)
k = 3

#get the current conditional power
cp <- getCP(n2, design$p1, design, k, mode = 1, alpha = 0.05)
cp


#increase n2 until the conditional power is larger than 0.8
while(cp < 0.8){
n2 <- n2 + 1
# Assume we spent the "rest alpha" proportionally (in the planning phase)
# therefore we set "mode = 1".
cp <- getCP(n2, design$p1, design, k, mode = 1, alpha = 0.05)
}

n2
```

getCP_simon *Returns the conditional power.*

Description

Returns the conditional power when "k" responses where observed out of "numPat" Patients for the given Simon's tow stage design.

Usage

```
getCP_simon(k, numPat, r1, n1, r, n, p1)
```

Arguments

k	number of observed responses
numPat	number of enrolled patients.
r1	critical value for the first stage.
n1	sample size for the first stage.
r	critical value for the subset endpoint.
n	overall sample size.
p1	response rate under the alternative hypothesis.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.
#Assume 3 out of 20 patients had a response.
getCP_simon(3,20,design$r1, design$n1, design$r, design$n, design$p1)
```

getD_distributeToOne *Get the conditional errors.*

Description

Calculates the conditional error for all possible outcomes at the interim analysis (different number of responses) spending the "rest alpha" (difference between nominal alpha level and actual alpha level) only to increase the worst case (smallest conditional error value that is not equal to 0).

Usage

```
getD_distributeToOne(design, alpha)
```

Arguments

design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums "r1", "n1", "r", "n" and "p0". • r1 = critical value for the first stage (more than r1 responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than r responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.
alpha	overall significance level the trial was planned for.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

See Also

getD_proportionally, getD_equally, getD_none

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

ce_toOne <- getD_distributeToOne(design, 0.05)
ce_toOne
```

---

getD_equally *Get the conditional errors equally.*

Description

Calculates the conditional error for all possible outcomes at the interim analysis (different number of responses) spending the "rest alpha" (difference between nominal alpha level and actual alpha level) equally.

Usage

```
getD_equally(design, alpha)
```

Arguments

design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums r1, n1, r, n and p0. • r1 = critical value for the first stage (more than r1 responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than r responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.
alpha	overall significance level the trial was planned for.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

See Also

getD_proportionally, getD_distributeToOne, getD_none

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

ce_equally <- getD_equally(design, 0.05)
ce_equally
```

---

getD_none *Get the conditional errors.*

Description

Calculates the conditional error for all possible outcomes at the interim analysis (different number of responses) using no "rest alpha" spending (difference between nominal alpha level and actual alpha level).

Usage

```
getD_none(design)
```

Arguments

design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums "r1", "n1", "r", "n" and "p0". • r1 = critical value for the first stage (more than r1 responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than r responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

See Also

getD_proportionally, getD_equally, getD_distributeToOne

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

ce_toOne <- getD_none(design)
ce_toOne
```

---

getD_proportionally *Get the conditional errors proportionally.*

Description

Calculates the conditional error for all possible outcomes at the interim analysis (different number of responses) spending "rest alpha" (difference between nominal alpha level and actual alpha level) proportionally.

Usage

```
getD_proportionally(design, alpha)
```

Arguments

design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums "r1", "n1", "r", "n" and "p0". • r1 = critical value for the first stage (more than "r1" responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than "r" responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.
alpha	overall significance level the trial was planned for.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

See Also

getD_equally, getD_distributeToOne, getD_none

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

ce_prop <- getD_proportionally(design, 0.05)
ce_prop
```

---

getN2 *Calculates the number of patients which should be enrolled in the second stage.*

Description

Calculates the number of patients which should be enrolled in the second stage if the conditional power should be altert to "cp".

Usage

```
getN2(cp, p1, design, k, mode = 0, alpha = 0.05)
```

Arguments

cp	conditional power to which the number of patients for the second stage should be adjusted.
p1	response probability under the alternative hypothesis.
design	a dataframe containing all critical values for a Simon's two-stage design defined by the colums r1, n1, r, n and p0. • r1 = critical value for the first stage (more than r1 responses needed to proceed to the second stage). • n1 = number of patients enrolled in the first stage. • r = critical value for the whole trial (more than r responses needed at the end of the study to reject the null hypothesis). • n = number of patients enrolled in the whole trial. • p0 = response probability under the null hypothesis.
k	number of responses observed at the interim analysis.
mode	a value out of {0,1,2,3} dedicating the methode spending the "rest alpha" (difference between nominal alpha level and actual alpha level for the given design). • 0 = "rest alpha" is not used. • 1 = "rest alpha" is spent proportionally. • 2 = "rest alpha" is spent equally. • 3 = "rest alpha" is spent only to the worst case scenario (minimal number of responses at the interim analysis so that the study can proceed to the second stage).
alpha	overall significance level the trial was planned for.

References

Englert S., Kieser M. (2012): Adaptive designs for single-arm phase II trials in oncology. Pharmaceutical Statistics 11,241-249.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

#Assume we only observed 3 responses in the interim analysis.
#Therefore the conditional power is only about 0.55.
#In order to raise the conditional power to 0.8 "n2" has to be increased.

#set k to 3 (only 3 responses observed so far)
k = 3

# Assume we spent the "rest alpha" proportionally in the planning phase
# there for we set "mode = 1".
n2 <- getN2(cp = 0.8, design$p1, design, k, mode = 1, alpha = 0.05)
n2
```

getP — *Calculates the p-value (binomial test).*

Description

Helper-function for the function `getCP`

Usage

```
getP(l, pi0, n2)
```

Arguments

`l`	number of responses
`pi0`	response probability under the null hypothesis
`n2`	number of enrolled patients

See Also

`getCP`

getSolutions — *Returns designs for a given "simon"-object (see `setupSimon`)*

Description

getSolutions uses a "simon"-object to calculate two-stage designs as they were described by Simon.

Usage

```
getSolutions(simon = setupSimon(), useCurtailment = FALSE,
  curtail_All = FALSE, cut = 0, replications = 10000, upperBorder = 0)
```

Arguments

`simon`	a "simon"-object which will be used to calculate designs.
`useCurtailment`	boolean value determining whether (non-)stochastic curtailment is used.
`curtail_All`	boolean value; if true the effect of (non-)stochastic curtailment will be calculated for different cut points in 0.05 steps starting with the value of the parameter "cut".
`cut`	sets the "cut point" used to calculate the effect of (non-)stochastic curtailment. A study is stopped if the conditional power falls below the value of "cut".
`replications`	number of simulations to estimate the effect of (non-)stochastic curtailment.
`upperBorder`	maximal possible value for n. If set to sero (default) the programm will aproximate a upper border automaticly.

References

Simon, R. (1989): Optimal two-stage designs for phase II clinical trials. Controlled Clinical Trials 10,1-10.

Kunz C.U., Kieser M (2012): Curtailment in single-arm two-stage phase II oncology trials. Biometrical Journal 54, 445-456

See Also

setupSimon

Examples

```
# Example 1: Using the default values
designs <- getSolutions()
designs <- designs$Solutions
designs

# Example 2: Setting up a "simon"-object, then calculate designs
simon <- setupSimon(alpha = 0.1, beta = 0.2, p0 = 0.3, p1 = 0.5)
designs <- getSolutions(simon)$Solutions
designs

# Esample 3: Calculating designs and simulating the influence of
# stochastic curtailment for each design.
simon <- setupSimon(alpha = 0.1, beta = 0.2, p0 = 0.3, p1 = 0.5)
designs <- getSolutions(simon, useCurtailment = TRUE, curtail_All = TRUE, cut = 0.3)
#List containing the found designs, the influence of stochastic curtailment
# and the regarding stopping rules.
designs
```

---

getSolutionsSub1 *Calculates designs for a given "sub1"-object.*

Description

By iterating over all possible values for "r1", "n1", "r", "s" and "n" designs for a given "sub1"-object are found. Proceeding to the second stage of the study more than "r1" responses among the first "n1" patients in the subset endpoint are needed. Rejecting the null hypothesis more than "r" responses in the subset endpoint or more than "s" responses in the superset endpoint among "n" patients are needed.

Usage

```
getSolutionsSub1(sub1 = setupSub1Design(), skipS = TRUE, skipR = TRUE,
  skipN1 = TRUE, lowerBorder = 0, upperBorder = 0,
  useCurtailment = FALSE, curtailAll = FALSE, cut = 0,
  replications = 1000)
```

Arguments

sub1	a "sub1"-object which will be used to calculate fitting designs
skipS	boolean value; skips the iteration over "s" at certian points to improve calculation speed (finds less designs)
skipR	boolean value; skips the iteration over "r" at certian points to improve calculation speed (finds less designs)
skipN1	boolean value; skips the iteration over "n1" at certian points to improve calculation speed (finds less designs and it is impossible to determine the optimalization criteria of the found designs)
lowerBorder	sets a minimal value for "n" (number of patients to be recruited)
upperBorder	sets a maximal value for "n" (number of patients to be recruited)
useCurtailment	determines if the effect of (non-)stochastic curtailment should also be calculated for the found designs
curtailAll	boolean value; if true the effect of (non-)stochastic curtailment will be calculated for different cut points in 0.05 steps starting with the value of the parameter "cut".
cut	sets the "cut point" used to calculate the effect of (non-)stochastic curtailment. A study is stopped if the conditional power falls below the value of "cut".
replications	number of simulations to estimate the effect of (non-)stochastic curtailment.

References

Kunz C.U., Kieser M (2012): Curtailment in single-arm two-stage phase II oncology trials. Biometrical Journal 54, 445-456

See Also

setupSub1Design

Examples

```
# Example 1: Using the default values
sub1 <- setupSub1Design()
getSolutionsSub1(sub1)

# Example 2: Setting up a "sub1"-object, then calculating designs
sub1 <- setupSub1Design(alpha = 0.1, beta = 0.2, pc0 = 0.3, pt0 = 0.4)
designs <- getSolutionsSub1(sub1)$Solutions
designs

# Example 2: Calculating designs and simulating the influence of stochastic curtailment
# for each design.
sub1 <- setupSub1Design(alpha = 0.1, beta = 0.2, pc0 = 0.3, pt0 = 0.4)
designs <- getSolutionsSub1(sub1, useCurtailment = TRUE, curtailAll = TRUE, cut = 0.3)
#Contains the found designs, the influence of stochastic curtailment
#and the regarding stopping rules .
designs
```

get_CI *Calculates the confidence interval.*

Description

Calculates the two sided 1-2*alpha confidence interval based on the work from Koyama and Chen.

Usage

```
get_CI(k, r1, n1, n, alpha = 0.05, precision = 4)
```

Arguments

k	overall observed responses (must be larger than r1).
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.
alpha	determining the two sided 1-2*alpha confidence interval.
precision	gives the precision (in decimal numbers) to which the confidence interval should be calculated (should be less than 10).

References

Koyama T and Chen H (2008): Proper inference from Simon's two-stage designs. Statistics in Medicine, 27(16):3145-3154.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

#Assume 9 responses were observed in the whole trial.
k = 9

ci <- get_CI(k, design$r1, design$n1, design$n)
```

get_conditionalPower *Calculates the conditional power.*

Description

Calculates the conditional power of a given subset design.

Usage

```
get_conditionalPower(t, u, enrolled, r1, n1, r, s, n, pc1, pt1,
  sub1 = setupSub1Design())
```

Arguments

t	observed responses in the subset endpoint.
u	observed responses in the superset endpoint.
enrolled	number of patients enrolled so far.
r1	critical value for the first stage.
n1	sample size for the first stage.
r	critical value for the subset endpoint.
s	critical value for the superset endpoint.
n	overall sample size.
pc1	the response probability under the alternative hypothesis for the subset endpoint.
pt1	the response probability under the alternative hypothesis for the superset endpoint.
sub1	"sub1"-object used to calculate the p value in c++ .

See Also

setupSub1Design

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a  subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

t <- 5
u <- 7
enrolled <- 10

con_p <- get_conditionalPower(t, u, enrolled, design$r1,
design$n1, design$r, design$s, design$n, design$pc1, design$pt1, sub1)
```

get_confidence_set *Calculates the confidence set.*

Description

The p-value of Subset Designs depends on two endpoints e.g. the superset endpoint and the subset endpoint. Therefore the confidence interval for the response rate of the subset endpoint depends on the response rate of the superset endpoint and vice versa. This results in a conficence "area" which is called the confidence set. "get_conficence_set" returns a set of points which outline the border of the confidence set.

Usage

```
get_confidence_set(t, u, r1, n1, n, pc0, pt0, alpha)
```

Arguments

t	observed responses in subset endpoint.
u	observed responses in the superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.
pc0	the response probability under the null hypothesis for the subset endpoint.
pt0	the response probability under the null hypothesis for the superset endpoint.
alpha	significance level the study was planned for.

References

Kunz, C. U. (2011), Two-stage designs for phase II trials with one or two endpoints. http://d-nb.info/1024218457

See Also

setupSub1Design, plot_confidence_set

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a  subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

t <- 12
u <- 13
alpha = 0.1

conf_set <- get_confidence_set(t, u, design$r1, design$n1, design$n, design$pc0, design$pt0, alpha)
```

get_p_exact_subset *Calculates the exact p value.*

Description

Calculates the exact p value for a given subset design.

Usage

```
get_p_exact_subset(t, u, r1, n1, n, pc0, pt0, sub1 = setupSub1Design())
```

Arguments

t	observed responses in the subset endpoint.
u	observed responses in the superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.
pc0	the response probability for the subset endpoint under the null hypothesis.
pt0	the response probability for the superset endpoint under the null hypothesis.
sub1	"sub1"-object used to calculate the p value in c++ .

See Also

setupSub1Design

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a  subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

#Assuming 9 responses in the subset endpoint and 13 responses
#in the superset endpoint were observed.
t = 9
u = 13

p_val <- get_p_exact_subset(t, u, design$r1, design$n1, design$n, design$pc0, design$pt0, sub1)
p_val
```

get_p_KC *Calculates the p-value.*

Description

Calculates the p-value for a Simon's two-stage design based on the work from Koyama and Chen.

Usage

```
get_p_KC(k, r1, n1, n, p0)
```

Arguments

k	overall observed responses.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.
p0	response probability under the null hypothesis.

References

Koyama T and Chen H (2008): Proper inference from Simon's two-stage designs. Statistics in Medicine, 27(16):3145-3154.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

#Assume 9 responses were observed in the whole trial.
k = 9

p_val <- get_p_KC(k, design$r1, design$n1, design$n, design$p0)
```

get_r2_flex *Calculates the number of responses needed for the second stage.*

Description

Calculates the number of responses needed for the second stage of a Simon's two-stage design if the flexible extension is chosen in the planning phase.

Usage

```
get_r2_flex(ce, p0, n2)
```

Arguments

ce	conditional error for the second stage.
p0	probability for a response under the null hypothesis.
n2	sample size for the second stage.

See Also

getD_proportionally, getD_equally, getD_distributeToOne, getD_none

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.
#Get the conditional error values using proportionally "rest"-alpha spending.
ce_df <- getD_proportionally(design, 0.05)
#Assume 5 responses were observed in the interim analysis.
ce <- ce_df[5+1,]$ce # conditional error for 5 responses is listed in the 6th row of "ce_df"
#Calculate the number of patients needed in the second stage.
n2 <- design$n - design$n1
r2 <- get_r2_flex(ce, design$p0, n2)
r2
#Assume 10 patients more should be recruited in the second stage.
#(This changes the number of needed responses.)
n2 <- n2 + 10
r2 <- get_r2_flex(ce, design$p0, n2)
r2
```

---

get_UMVUE_GMS *Calculates the "uniformly minimal variance unbiased estimator".*

Description

Calculates the "uniformly minimal variance unbiased estimator" (UMVUE) for the true response rate based on the approach of Grishick, Mosteller and Savage.

Usage

```
get_UMVUE_GMS(k, r1, n1, n)
```

Arguments

k	overall observed responses.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.

References

Girshick MA, Mosteller F, and Savage LJ (1946): Unbiased estimates for certain binomial sampling problems with applications. Annals of Mathematical Statistics, 17(1):13-23.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.

#Assume 9 responses were observed in the whole trial.
k = 9

umvue <- get_UMVUE_GMS(k, design$r1, design$n1, design$n)
```

---

get_UMVUE_GMS_subset_second_only

Calculates the "uniformly minimal variance unbiased estimator".

Description

Calculates the "uniformly minimal variance unbiased estimator" (UMVUE) for the true response rate only for the superset endpoint (response rate superset endpoint minus response rate subset endpoint) in a subset design.

Usage

```
get_UMVUE_GMS_subset_second_only(t, u, r1, n1, n)
```

Arguments

t	observed responses in the subset endpoint.
u	observed responses in the superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

#Assume 9 responses in the subset endpoint and 13 responses in the superset endpoint were observed.
t = 9
u = 13
umvue_second <- get_UMVUE_GMS_subset_second_only(t, u, design$r1, design$n1, design$n)
```

get_UMVUE_GMS_subset_second_total

Calculates the "uniformly minimal variance unbiased estimator".

Description

Calculates the "uniformly minimal variance unbiased estimator" (UMVUE) for the true response rate for the superset endpoint.

Usage

```
get_UMVUE_GMS_subset_second_total(t, u, r1, n1, n)
```

Arguments

t	observed responses in the subset endpoint.
u	observed responses in the superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

#Assume 9 responses in the subset endpoint and 13 responses in the superset endpoint were observed.
t = 9
u = 13
umvue_second <- get_UMVUE_GMS_subset_second_total(t, u, design$r1, design$n1, design$n)
```

plot_confidence_set

Plots the "confidence set" according to the observed responses.

Description

Plots the "confidence set" which can be received by invoking "get_confidence_set". Also the "uniformly minimal variance unbiased estimator" and the acceptance area are included in the plot.

Usage

```
plot_confidence_set(t, u, r1, n1, n, pc0, pt0, alpha)
```

Arguments

t	observed responses in the subset endpoint.
u	observed responses in the superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.
n	overall sample size.
pc0	the response probability for the subset endpoint under the null hypothesis.
pt0	the response probability for the superset endpoint under the null hypothesis.
alpha	overall significance level the trial was planned for.

References

Kunz, C. U. (2011), Two-stage designs for phase II trials with one or two endpoints. http://d-nb.info/1024218457

See Also

get_confidence_set, get_UMVUE_GMS_subset_second_total, get_UMVUE_GMS

Examples

```
#Setup "sub1"-object
sub1 <- setupSub1Design(pc0 = 0.5, pt0 = 0.6)

#Calculate a  subset design
design <- getSolutionsSub1(sub1, skipN1 = FALSE)$Solutions[4,]

#Assume 11 responses in the subset endpoint and 12 responses in the superset endpoint were observed.
t = 10
u = 12

plot_confidence_set(t, u, design$r1, design$n1, design$n, design$pc0, design$pt0, 0.1)
```

---

plot_simon_study_state

Plots the study state of a given Simon's two-stage design.

Description

Plots the study state of a given Simon's two-stage design displaying the already enrolled patients and the stopping rules.

Usage

```
plot_simon_study_state(sr, enrolledPat = data.frame(ep1 = logical()), r1, n1,
  r, n)
```

Arguments

sr	dataframe containing the stopping rules for the given Simon's two-stage design defined by two columns named "Enrolled_patients" and "Needed_responses_ep1". This way each row defines when the study has to be stopped for futility.
enrolledPat	dataframe defined by a boolean vector in one column named "ep1" indicating which patient had a response.
r1	critical value for the first stage.
n1	sample size for the first stage.
r	critical value for the subset endpoint.
n	overall sample size.

Examples

```
#Calculate a Simon's two-stage design
design <- getSolutions()$Solutions[3,] #minimax-design for the default values.
#Define the stopping rules according to the chosen design
sr <- data.frame(Enrolled_patients = c(design$n1, design$n),
Needed_responses_ep1 = c(design$r1, design$r))
#Simulate 18 random generated outcomes.
enrolledPat <- data.frame(ep1 = rbinom(18,1, design$p1))
#Plot study state
plot_simon_study_state(sr, enrolledPat, design$r1, design$n1, design$r, design$n)
```

---

plot_sub1_study_state *Plots the study state of a given subset design.*

Description

Plots the study state of a given subset design displaying the already enrolled patients and the stopping rules for the given study.

Usage

```
plot_sub1_study_state(sr, enrolledPat = data.frame(ep1 = logical(), ep2 =
  logical()), r1, n1, r, s, n)
```

Arguments

sr	dataframe containing the stopping rules for the given subset design defined by 3 columns named "Enrolled_patients", "Needed_responses_ep1" and "Needed_responses_ep2". This way each row defines when the study has to be stopped for futility.
enrolledPat	dataframe defined by two boolean vectors named "ep1" and "ep2" indicating which patient had a response in the subset and superset endpoint.
r1	critical value for the first stage.
n1	sample size for the first stage.

r	critical value for the subset endpoint.
s	critical value for the superset endpoint.
n	overall sample size.

See Also

getSolutionsSub1

Examples

```
#Calculate a subset design.
sub1 <- setupSub1Design(alpha = 0.1, beta = 0.2, pc0 = 0.3, pt0 = 0.4)
design <- getSolutionsSub1(sub1)$Solutions[10,]
#Define the stopping rules according to the chosen design.
sr <- data.frame(Enrolled_patients = c(design$n1, design$n),
Needed_responses_ep1 = c(design$r1, design$r), Needed_responses_ep2 = c(0,design$s))
#Simulate 14 random generated outcomes.
tmp_ep1 <- rbinom(14,1, design$pc1)
tmp_ep2 <- tmp_ep1 | rbinom(14,1, design$pt1)
enrolledPat <- data.frame(ep1 = tmp_ep1, ep2 = tmp_ep2)
#Plot study state.
plot_sub1_study_state(sr, enrolledPat, design$r1, design$n1, design$r, design$s, design$n)
```

Rcpp Modules simon — *Functions and Objects created by Rcpp for the "Simon's two-stage design"*

Description

The functions and objects of the "simon module" are accessible from R via the Rcpp Modules mechanism which creates them based on the declaration in the C++ files. The whole implemented functionality of the "simon module" is used by the functions implemented in this package. Therefore there is no need for the user to access the functions of the module directly.

References

https://CRAN.R-project.org/package=Rcpp

See Also

setupSimon, getSolutions

Rcpp Modules sub1 *Functions and Objects created by Rcpp for the "subset design"*

Description

The functions and objects of the "sub1 module" are accessible from R via the Rcpp Modules mechanism which creates them based on the declaration in the C++ files. The whole implemented functionality of the "sub1 module" is used by the functions implemented in this package. Therefore there is no need for the user to access the functions of the module directly.

References

https://CRAN.R-project.org/package=Rcpp

See Also

setupSub1Design, getSolutionsSub1

setSimonParams *Sets the parameters for a given "simon"-object.*

Description

Sets the parameters for a given "simon"-object.

Usage

```
setSimonParams(s, alpha = 0.05, beta = 0.05, p0 = 0.1, p1 = 0.3)
```

Arguments

s	a "simon"-object which is generated by the function setupSimon.
alpha	the maximal type I error rate.
beta	the maximal type II error rate.
p0	the response probability under the null hypothesis.
p1	the response probability under the alternative hypothesis.

See Also

setupSimon

Examples

```
#Create "simon"-object.
simon <- setupSimon()
#Change parameters.
setSimonParams(simon, alpha = 0.1, beta = 0.2, p0 = 0.25, p1 = 0.45)
#Calculate designs for the given "simon"-object.
designs <- getSolutions(simon)$Solutions
designs
```

---

setSub1Params *Sets the parameters for a given "sub1"-object.*

Description

Sets the parameters for a given "sub1"-object.

Usage

```
setSub1Params(sub1, alpha = 0.1, beta = 0.1, pc0 = 0.6, pt0 = 0.7,
  pc1 = 0.8, pt1 = 0.9)
```

Arguments

sub1	a "sub1"-object which is generated by the function setupSub1Design.
alpha	the maximal type I error rate.
beta	the maximal type II error rate.
pc0	the response probability for the subset endpoint under the null hypothesis.
pt0	the response probability for the superset endpoint under the null hypothesis.
pc1	the response probability for the subset endpoint under the alternative hypothesis.
pt1	the response probability for the superset endpoint under the alternative hypothesis.

See Also

setupSub1Design

Examples

```
#Create "sub1"-object.
sub1 <- setupSub1Design()
#Change parameters.
setSub1Params(sub1, beta = 0.2, pc0 = 0.5, pt0 = 0.6)
#Calculate designs for the given "sub1"-object.
designs <- getSolutionsSub1(sub1)$Solutions
designs
```

setupSimon *Creates a "simon"-object.*

Description

Creates a "simon"-object which can be used in the function `getSolutions` to identify possible designs.

Usage

```
setupSimon(alpha = 0.05, beta = 0.05, p0 = 0.1, p1 = 0.3)
```

Arguments

`alpha`	the maximal type I error rate.
`beta`	the maximal type II error rate.
`p0`	the response probability under the null hypothesis.
`p1`	the response probability under the alternative hypothesis.

Examples

```
#Create a "simon"-object
simon <- setupSimon()
#Calculate designs for the given "simon"-object.
designs <- getSolutions(simon)$Solutions
designs
```

setupSub1Design *Creates a "sub1"-object.*

Description

Creates a "sub1"-object which can be used in the function `getSolutionsSub1` to identify possible designs.

Usage

```
setupSub1Design(alpha = 0.1, beta = 0.1, pc0 = 0.6, pt0 = 0.7,
  pc1 = 0.8, pt1 = 0.9)
```

Arguments

alpha	the maximal type I error rate.
beta	the maximal type II error rate.
pc0	the response probability for the subset endpoint under the null hypothesis.
pt0	the response probability for the superset endpoint under the null hypothesis.
pc1	the response probability for the subset endpoint under the alternative hypothesis.
pt1	the response probability for the superset endpoint under the alternative hypothesis.

Examples

```
#Create "sub1"-object.
sub1 <- setupSub1Design()
#Calculate designs for the given "sub1"-object.
designs <- getSolutionsSub1(sub1)$Solutions
designs
```

---

toDataframe *Helper function for* getSolutions *and* getSolutionsSub1.

Description

Transfers the found designs of a "design"-object to a dataframe. This function is a helper-function used in getSolutions and getSolutionsSub1 to structure the found designs for a given parameter set specified in a "sub1"-object or "simon"-object. The return value is a list containing a dataframe for the found designs and multiple dataframes containing the results for (non-)stochastic curtailment, if present.

Usage

```
toDataframe(designObject, useCurtailment = F)
```

Arguments

designObject	either a "sub1"-object or a "simon"-object containing the designs for a given parameter set specified inside the "designObject".
useCurtailment	boolean value dedicating the use of (non-)stochastic curtailment. This parameter indicates if that information should also gathered.

Index