Package 'rpact'

March 17, 2026

Title Confirmatory Adaptive Clinical Trial Design and Analysis

Version 4.4.0

Date 2026-03-04

Description Design and analysis of confirmatory adaptive clinical trials with continuous, binary, and survival endpoints according to the methods described in the monograph by Wassmer and Brannath (2025) <doi:10.1007/978-3-031-89669-9>. This includes classical group sequential as well as multi-stage adaptive hypotheses tests that are based on the combination testing principle.

License LGPL-3

Encoding UTF-8

LazyData true

URL https://www.rpact.org,
https://www.rpact.com,
https://docs.rpact.org,
https://github.com/rpact-com/rpact,
https://rpact-cloud.share.connect.posit.cloud

BugReports https://github.com/rpact-com/rpact/issues

Language en-US

Depends R (>= 3.6.0)

Imports methods,
stats,
utils,
graphics,
tools,
rlang,
R6 (>= 2.5.1),
knitr (>= 1.19),
Rcpp (>= 1.0.3)

LinkingTo Rcpp

Suggests ggplot2 (>= 3.5.0),
testthat (>= 3.0.0),
rmarkdown (>= 1.10),
rappdirs (>= 0.3.3)

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first *analysis*

Collate 'RcppExports.R'
'f_logger.R'
'class_dictionary.R'
'f_core_constants.R'
'f_core_utilities.R'
'f_core_assertions.R'
'f_analysis_utilities.R'
'f_parameter_set_utilities.R'
'class_core_parameter_set.R'
'class_core_plot_settings.R'
'f_core_plot.R'
'class_design.R'
'f_object_r_code.R'
'f_analysis_base.R'
'class_analysis_dataset.R'
'class_analysis_stage_results.R'
'class_analysis_results.R'
'f_design_general_utilities.R'
'class_time.R'
'class_design_set.R'
'class_design_plan.R'
'class_design_power_and_asn.R'
'class_event_probabilities.R'
'f_simulation_base_counts.R'
'f_simulation_utilities.R'
'f_simulation_base_survival.R'
'class_simulation_results.R'
'class_performance_score.R'
'class_summary.R'
'data.R'
'f_analysis_base_means.R'
'f_analysis_base_rates.R'
'f_analysis_base_survival.R'
'f_analysis_boundary_recalculation.R'
'f_analysis_enrichment.R'
'f_analysis_enrichment_means.R'
'f_analysis_enrichment_rates.R'
'f_analysis_enrichment_survival.R'
'f_analysis_multiarm.R'
'f_analysis_multiarm_means.R'
'f_analysis_multiarm_rates.R'
'f_analysis_multiarm_survival.R'
'f_as251.R'
'f_core_output_formats.R'
'f_design_fisher_combination_test.R'
'f_design_futility_bounds.R'

'f_design_group_sequential.R'
'f_design_plan_counts.R'
'f_design_plan_means.R'
'f_design_plan_plot.R'
'f_design_plan_rates.R'
'f_design_plan_survival.R'
'f_design_plan_utilities.R'
'f_io_utilities.R'
'f_quality_assurance.R'
'f_simulation_base_means.R'
'f_simulation_base_rates.R'
'f_simulation_calc_subjects_function.R'
'f_simulation_enrichment.R'
'f_simulation_enrichment_means.R'
'f_simulation_enrichment_rates.R'
'f_simulation_enrichment_survival.R'
'f_simulation_multiarm.R'
'f_simulation_multiarm_means.R'
'f_simulation_multiarm_rates.R'
'f_simulation_multiarm_survival.R'
'f_simulation_performance_score.R'
'f_simulation_plot.R'
'parameter_descriptions.R'
'pkgname.R'

Contents

`AccrualTime` *Accrual Time*

Description

Class for the definition of accrual time and accrual intensity.

Details

`AccrualTime` is a class for the definition of accrual time and accrual intensity.

Fields

`endOfAccrualIsUserDefined` If `TRUE`, the end of accrual has to be defined by the user (i.e., the length of `accrualTime` is equal to the length of `accrualIntensity` -1). Is a logical vector of length 1.

`followUpTimeMustBeUserDefined` Specifies whether follow up time needs to be defined or not. Is a logical vector of length 1.

`maxNumberOfSubjectsIsUserDefined` If `TRUE`, the maximum number of subjects has been specified by the user, if `FALSE`, it was calculated.

`maxNumberOfSubjectsCanBeCalculatedDirectly` If `TRUE`, the maximum number of subjects can directly be calculated. Is a logical vector of length 1.

`absoluteAccrualIntensityEnabled` If `TRUE`, absolute accrual intensity is enabled. Is a logical vector of length 1.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`accrualIntensityRelative` The relative accrual intensities.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`remainingTime` In survival designs, the remaining time for observation. Is a numeric vector of length 1.

`piecewiseAccrualEnabled` Indicates whether piecewise accrual is selected. Is a logical vector of length 1.

`AnalysisResults` *Basic Class for Analysis Results*

Description

A basic class for analysis results.

Details

`AnalysisResults` is the basic class for

- `AnalysisResultsFisher`,
- `AnalysisResultsGroupSequential`,
- `AnalysisResultsInverseNormal`,
- `AnalysisResultsMultiArmFisher`,
- `AnalysisResultsMultiArmInverseNormal`,
- `AnalysisResultsConditionalDunnett`,
- `AnalysisResultsEnrichmentFisher`,
- `AnalysisResultsEnrichmentInverseNormal`.

AnalysisResultsConditionalDunnett

Analysis Results Multi-Arm Conditional Dunnett

Description

Class for multi-arm analysis results based on a conditional Dunnett test design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the multi-arm analysis results of a conditional Dunnett test design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`piControl` The assumed probability in the control arm for simulation and under which the sample size recalculation is performed. Is a numeric vector of length 1 containing a value between 0 and 1.

`AnalysisResultsEnrichment`

Basic Class for Analysis Results Enrichment

Description

A basic class for enrichment analysis results.

Details

`AnalysisResultsEnrichment` is the basic class for

- `AnalysisResultsEnrichmentFisher` and
- `AnalysisResultsEnrichmentInverseNormal`.

`AnalysisResultsEnrichmentFisher`

Analysis Results Enrichment Fisher

Description

Class for enrichment analysis results based on a Fisher combination test design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the multi-arm analysis results of a Fisher combination test design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`piControls` The assumed rates in the control group for enrichment designs, i.e., designs with multiple subsets.

`conditionalPowerSimulated` The simulated conditional power, under the assumption of observed or assumed effect sizes.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

`AnalysisResultsEnrichmentInverseNormal`

Analysis Results Enrichment Inverse Normal

Description

Class for enrichment analysis results based on a inverse normal design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the enrichment analysis results of an inverse normal design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`piControls` The assumed rates in the control group for enrichment designs, i.e., designs with multiple subsets.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

AnalysisResultsFisher *Analysis Results Fisher*

Description

Class for analysis results based on a Fisher combination test design.

Details

This object cannot be created directly; use getAnalysisResults with suitable arguments to create the analysis results of a Fisher combination test design.

Fields

normalApproximation Describes if a normal approximation was used when calculating p-values. Default for means is FALSE and TRUE for rates and hazard ratio. Is a logical vector of length 1.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

thetaH0 The difference or assumed effect under H0. Is a numeric vector of length 1.

pi1 The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

pi2 The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

nPlanned The sample size planned for each of the subsequent stages. Is a numeric vector of length kMax containing whole numbers.

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

thetaH1 The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

assumedStDev The assumed standard deviation(s) for means analysis. Is a numeric vector.

equalVariances Describes if the variances in two treatment groups are assumed to be the same. Is a logical vector of length 1.

testActions The test decisions at each stage of the trial. Is a character vector of length kMax.

conditionalRejectionProbabilities The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length kMax containing values between 0 and 1.

conditionalPower The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

repeatedConfidenceIntervalLowerBounds The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length kMax.

repeatedConfidenceIntervalUpperBounds The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length kMax.

repeatedPValues The p-values that are calculated at any stage of the trial. Is a numeric vector of length kMax containing values between 0 and 1.

`finalStage` The stage at which the trial ends, either with acceptance or rejection of the null hypothesis. Is a numeric vector of length 1.

`finalPValues` The final p-value that is based on the stage-wise ordering. Is a numeric vector of length `kMax` containing values between 0 and 1.

`finalConfidenceIntervalLowerBounds` The lower bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length `kMax`.

`finalConfidenceIntervalUpperBounds` The upper bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length `kMax`.

`medianUnbiasedEstimates` The calculated median unbiased estimates that are based on the stage-wise ordering. Is a numeric vector of length `kMax`.

`conditionalPowerSimulated` The simulated conditional power, under the assumption of observed or assumed effect sizes.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`AnalysisResultsGroupSequential`

Analysis Results Group Sequential

Description

Class for analysis results results based on a group sequential design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the analysis results of a group sequential design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

thetaH1 The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

assumedStDev The assumed standard deviation(s) for means analysis. Is a numeric vector.

equalVariances Describes if the variances in two treatment groups are assumed to be the same. Is a logical vector of length 1.

testActions The test decisions at each stage of the trial. Is a character vector of length kMax.

conditionalRejectionProbabilities The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length kMax containing values between 0 and 1.

conditionalPower The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

repeatedConfidenceIntervalLowerBounds The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length kMax.

repeatedConfidenceIntervalUpperBounds The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length kMax.

repeatedPValues The p-values that are calculated at any stage of the trial. Is a numeric vector of length kMax containing values between 0 and 1.

finalStage The stage at which the trial ends, either with acceptance or rejection of the null hypothesis. Is a numeric vector of length 1.

finalPValues The final p-value that is based on the stage-wise ordering. Is a numeric vector of length kMax containing values between 0 and 1.

finalConfidenceIntervalLowerBounds The lower bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length kMax.

finalConfidenceIntervalUpperBounds The upper bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length kMax.

medianUnbiasedEstimates The calculated median unbiased estimates that are based on the stage-wise ordering. Is a numeric vector of length kMax.

maxInformation The maximum information. Is a numeric vector of length 1 containing a whole number.

informationEpsilon The absolute information epsilon, which defines the maximum distance from the observed information to the maximum information that causes the final analysis. Updates at the final analysis if the observed information at the final analysis is smaller ("underrunning") than the planned maximum information. Is either a positive integer value specifying the absolute information epsilon or a floating point number >0 and <1 to define a relative information epsilon.

AnalysisResultsInverseNormal

Analysis Results Inverse Normal

Description

Class for analysis results results based on an inverse normal design.

Details

This object cannot be created directly; use getAnalysisResults with suitable arguments to create the analysis results of a inverse normal design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDev` The assumed standard deviation(s) for means analysis. Is a numeric vector.

`equalVariances` Describes if the variances in two treatment groups are assumed to be the same. Is a logical vector of length 1.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`finalStage` The stage at which the trial ends, either with acceptance or rejection of the null hypothesis. Is a numeric vector of length 1.

`finalPValues` The final p-value that is based on the stage-wise ordering. Is a numeric vector of length `kMax` containing values between 0 and 1.

`finalConfidenceIntervalLowerBounds` The lower bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length `kMax`.

`finalConfidenceIntervalUpperBounds` The upper bound of the confidence interval that is based on the stage-wise ordering. Is a numeric vector of length `kMax`.

`medianUnbiasedEstimates` The calculated median unbiased estimates that are based on the stage-wise ordering. Is a numeric vector of length `kMax`.

AnalysisResultsMultiArm
Basic Class for Analysis Results Multi-Arm

Description

A basic class for multi-arm analysis results.

Details

`AnalysisResultsMultiArm` is the basic class for

- `AnalysisResultsMultiArmFisher`,
- `AnalysisResultsMultiArmInverseNormal`, and
- `AnalysisResultsConditionalDunnett`.

AnalysisResultsMultiArmFisher
Analysis Results Multi-Arm Fisher

Description

Class for multi-arm analysis results based on a Fisher combination test design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the multi-arm analysis results of a Fisher combination test design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`piControl` The assumed probability in the control arm for simulation and under which the sample size recalculation is performed. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerSimulated` The simulated conditional power, under the assumption of observed or assumed effect sizes.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`AnalysisResultsMultiArmInverseNormal`
Analysis Results Multi-Arm Inverse Normal

Description

Class for multi-arm analysis results based on a inverse normal design.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the multi-arm analysis results of an inverse normal design.

Fields

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`conditionalRejectionProbabilities` The probabilities of rejecting the null hypothesis at each stage, given the stage has been reached. Is a numeric vector of length `kMax` containing values between 0 and 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`repeatedConfidenceIntervalLowerBounds` The lower bound of the confidence intervals that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedConfidenceIntervalUpperBounds` The upper bound of the confidence interval that are calculated at any stage of the trial. Is a numeric vector of length `kMax`.

`repeatedPValues` The p-values that are calculated at any stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`piControl` The assumed probability in the control arm for simulation and under which the sample size recalculation is performed. Is a numeric vector of length 1 containing a value between 0 and 1.

AnalysisResultsMultiHypotheses

Basic Class for Analysis Results Multi-Hypotheses

Description

A basic class for multi-hypotheses analysis results.

Details

`AnalysisResultsMultiHypotheses` is the basic class for

- `AnalysisResultsMultiArm` and
- `AnalysisResultsEnrichment`.

as.data.frame.AnalysisResults

Coerce AnalysisResults to a Data Frame

Description

Returns the `AnalysisResults` object as data frame.

Usage

```
## S3 method for class 'AnalysisResults'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  niceColumnNamesEnabled = FALSE
)
```

Arguments

`x`	An `AnalysisResults` object created by `getAnalysisResults()`.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`niceColumnNamesEnabled`	Logical. If `TRUE`, nice looking column names will be used; syntactic names (variable names) otherwise (see `make.names`).

Details

Coerces the analysis results to a data frame.

Value

Returns a `data.frame`.

as.data.frame.ParameterSet
Coerce Parameter Set to a Data Frame

Description

Returns the ParameterSet as data frame.

Usage

```
## S3 method for class 'ParameterSet'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE
)
```

Arguments

x	A FieldSet object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.

Details

Coerces the parameter set to a data frame.

Value

Returns a data.frame.

as.data.frame.PowerAndAverageSampleNumberResult
Coerce Power And Average Sample Number Result to a Data Frame

Description

Returns the PowerAndAverageSampleNumberResult as data frame.

Usage

```
## S3 method for class 'PowerAndAverageSampleNumberResult'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  ...
)
```

Arguments

x	A PowerAndAverageSampleNumberResult object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Coerces the PowerAndAverageSampleNumberResult object to a data frame.

Value

Returns a data.frame.

Examples

```
## Not run:
data <- as.data.frame(getPowerAndAverageSampleNumber(getDesignGroupSequential()))
head(data)
dim(data)

## End(Not run)
```

---

as.data.frame.StageResults

Coerce Stage Results to a Data Frame

Description

Returns the StageResults as data frame.

Usage

```
## S3 method for class 'StageResults'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  type = 1,
  ...
)
```

Arguments

x	A StageResults object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Coerces the stage results to a data frame.

Value

Returns a data.frame.

---

as.data.frame.TrialDesign

Coerce TrialDesign to a Data Frame

Description

Returns the TrialDesign as data frame.

Usage

```
## S3 method for class 'TrialDesign'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  ...
)
```

Arguments

x	A TrialDesign object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Each element of the TrialDesign is converted to a column in the data frame.

Value

Returns a data.frame.

Examples

```
## Not run:
as.data.frame(getDesignGroupSequential())

## End(Not run)
```

---

as.data.frame.TrialDesignCharacteristics

Coerce TrialDesignCharacteristics to a Data Frame

Description

Returns the TrialDesignCharacteristics as data frame.

Usage

```
## S3 method for class 'TrialDesignCharacteristics'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  ...
)
```

Arguments

x	A TrialDesignCharacteristics object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Each element of the TrialDesignCharacteristics is converted to a column in the data frame.

Value

Returns a data.frame.

Examples

```
## Not run:
as.data.frame(getDesignCharacteristics(getDesignGroupSequential()))

## End(Not run)
```

---

as.data.frame.TrialDesignPlan

Coerce Trial Design Plan to a Data Frame

Description

Returns the TrialDesignPlan as data frame.

Usage

```
## S3 method for class 'TrialDesignPlan'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  ...
)
```

Arguments

x	A TrialDesignPlan object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Coerces the design plan to a data frame.

Value

Returns a data.frame.

Examples

```
## Not run:
as.data.frame(getSampleSizeMeans())

## End(Not run)
```

---

as.data.frame.TrialDesignSet

Coerce Trial Design Set to a Data Frame

Description

Returns the TrialDesignSet as data frame.

Usage

```
## S3 method for class 'TrialDesignSet'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  niceColumnNamesEnabled = FALSE,
  includeAllParameters = FALSE,
  addPowerAndAverageSampleNumber = FALSE,
  theta = seq(-1, 1, 0.02),
  nMax = NA_integer_,
  ...
)
```

Arguments

x	A TrialDesignSet object.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).
includeAllParameters	Logical. If TRUE, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is FALSE.
addPowerAndAverageSampleNumber	If TRUE, power and average sample size will be added to data frame, default is FALSE.
theta	A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.
nMax	The maximum sample size. Must be a positive integer of length 1.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Coerces the design set to a data frame.

Value

Returns a data.frame.

Examples

```
## Not run:
designSet <- getDesignSet(design = getDesignGroupSequential(), alpha = c(0.01, 0.05))
as.data.frame(designSet)

## End(Not run)
```

---

as.matrix.FieldSet *Coerce Field Set to a Matrix*

Description

Returns the FrameSet as matrix.

Usage

```
## S3 method for class 'FieldSet'
as.matrix(x, ..., enforceRowNames = TRUE, niceColumnNamesEnabled = TRUE)
```

Arguments

x	A FieldSet object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
enforceRowNames	If TRUE, row names will be created depending on the object type, default is TRUE.
niceColumnNamesEnabled	Logical. If TRUE, nice looking column names will be used; syntactic names (variable names) otherwise (see make.names).

Details

Coerces the frame set to a matrix.

Value

Returns a matrix.

---

as251Normal *Algorithm AS 251: Normal Distribution*

Description

Calculates the Multivariate Normal Distribution with Product Correlation Structure published by Charles Dunnett, Algorithm AS 251.1 Appl.Statist. (1989), Vol.38, No.3, doi:10.2307/2347754.

Usage

```
as251Normal(
  lower,
  upper,
  sigma,
  ...,
  eps = 1e-06,
  errorControl = c("strict", "halvingIntervals"),
  intervalSimpsonsRule = 0
)
```

Arguments

lower	Lower limits of integration. Array of N dimensions
upper	Upper limits of integration. Array of N dimensions
sigma	Values defining correlation structure. Array of N dimensions
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
eps	desired accuracy. Defaults to 1e-06
errorControl	error control. If set to 1, strict error control based on fourth derivative is used. If set to zero, error control based on halving intervals is used
intervalSimpsonsRule	Interval width for Simpson's rule. Value of zero caused a default .24 to be used

Details

For a multivariate normal vector with correlation structure defined by rho(i,j) = bpd(i) * bpd(j), computes the probability that the vector falls in a rectangle in n-space with error less than eps.

This function calculates the bdp value from `sigma`, determines the right `inf` value and calls `mvnprd`.

`as251StudentT` *Algorithm AS 251: Student T Distribution*

Description

Calculates the Multivariate Normal Distribution with Product Correlation Structure published by Charles Dunnett, Algorithm AS 251.1 Appl.Statist. (1989), Vol.38, No.3, doi:10.2307/2347754.

Usage

```
as251StudentT(
  lower,
  upper,
  sigma,
  ...,
  df,
  eps = 1e-06,
  errorControl = c("strict", "halvingIntervals"),
  intervalSimpsonsRule = 0
)
```

Arguments

`lower`	Lower limits of integration. Array of N dimensions
`upper`	Upper limits of integration. Array of N dimensions
`sigma`	Values defining correlation structure. Array of N dimensions
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`df`	Degrees of Freedom. Use 0 for infinite D.F.
`eps`	desired accuracy. Defaults to 1e-06
`errorControl`	error control. If set to 1, strict error control based on fourth derivative is used. If set to zero, error control based on halving intervals is used
`intervalSimpsonsRule`	Interval width for Simpson's rule. Value of zero caused a default .24 to be used

Details

For a multivariate normal vector with correlation structure defined by rho(i,j) = bpd(i) * bpd(j), computes the probability that the vector falls in a rectangle in n-space with error less than eps.

This function calculates the bdp value from `sigma`, determines the right `inf` value and calls `mvstud`.

checkInstallationQualificationStatus
Check Installation Qualification Status

Description

This function checks whether the installation qualification for the `rpact` package has been completed. If not, it provides a message prompting the user to run the `testPackage()` function to perform the qualification.

Usage

```
checkInstallationQualificationStatus(showMessage = TRUE)
```

Arguments

`showMessage`	A logical value indicating whether to display a message if the installation qualification has not been completed. Default is `TRUE`.

Details

The installation qualification is a critical step in ensuring that the `rpact` package is correctly installed and validated for use in GxP-relevant environments. This function verifies the qualification status and informs the user if further action is required.

Value

Invisibly returns `TRUE` if the installation qualification has been completed, otherwise returns `FALSE`.

Examples

```
## Not run:
checkInstallationQualificationStatus()

## End(Not run)
```

ClosedCombinationTestResults
Analysis Results Closed Combination Test

Description

Class for multi-arm analysis results based on a closed combination test.

Details

This object cannot be created directly; use `getAnalysisResults` with suitable arguments to create the multi-arm analysis results of a closed combination test design.

Fields

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`indices` Indicates which stages are available for analysis.

`adjustedStageWisePValues` The multiplicity adjusted p-values from the separate stages. Is a numeric matrix.

`overallAdjustedTestStatistics` The overall adjusted test statistics.

`separatePValues` The p-values from the separate stages. Is a numeric matrix.

`conditionalErrorRate` The calculated conditional error rate.

`secondStagePValues` For conditional Dunnett test, the conditional or unconditional p-value calculated for the second stage.

`rejected` Indicates whether a hypothesis is rejected or not.

`rejectedIntersections` The simulated number of rejected arms in the closed testing procedure.. Is a logical matrix.

`ConditionalPowerResults` *Conditional Power Results*

Description

Class for conditional power calculations

Details

This object cannot be created directly; use `getConditionalPower()` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDev` The assumed standard deviation(s) for means analysis. Is a numeric vector.

ConditionalPowerResultsEnrichmentMeans

Conditional Power Results Enrichment Means

Description

Class for conditional power calculations of enrichment means data

Details

This object cannot be created directly; use `getConditionalPower` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDevs` Assumed standard deviations to calculate conditional power in multi-arm trials or enrichment designs. Is a numeric vector.

ConditionalPowerResultsEnrichmentRates

Conditional Power Results Enrichment Rates

Description

Class for conditional power calculations of enrichment rates data

Details

This object cannot be created directly; use `getConditionalPower` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`piTreatments` The assumed rates in the treatment groups for multi-arm and enrichment designs, i.e., designs with multiple subsets.

`piControls` The assumed rates in the control group for enrichment designs, i.e., designs with multiple subsets.

`ConditionalPowerResultsMeans`

Conditional Power Results Means

Description

Class for conditional power calculations of means data

Details

This object cannot be created directly; use `getConditionalPower` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`assumedStDev` The assumed standard deviation(s) for means analysis. Is a numeric vector.

ConditionalPowerResultsRates

Conditional Power Results Rates

Description

Class for conditional power calculations of rates data

Details

This object cannot be created directly; use `getConditionalPower` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

ConditionalPowerResultsSurvival

Conditional Power Results Survival

Description

Class for conditional power calculations of survival data

Details

This object cannot be created directly; use `getConditionalPower` with suitable arguments to create the results of a group sequential or a combination test design.

Fields

`nPlanned` The sample size planned for each of the subsequent stages. Is a numeric vector of length `kMax` containing whole numbers.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`simulated` Describes if the power for Fisher's combination test has been simulated. Only applicable when using Fisher designs. Is a logical vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`dataEnrichmentMeans` *Enrichment Dataset of Means*

Description

A dataset containing the sample sizes, means, and standard deviations of two groups. Use `getDataset(dataEnrichment` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentMeans
```

Format

A `data.frame` object.

`dataEnrichmentMeansStratified` *Stratified Enrichment Dataset of Means*

Description

A dataset containing the sample sizes, means, and standard deviations of two groups. Use `getDataset(dataEnrichment` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentMeansStratified
```

Format

A `data.frame` object.

dataEnrichmentRates *Enrichment Dataset of Rates*

Description

A dataset containing the sample sizes and events of two groups. Use `getDataset(dataEnrichmentRates)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentRates
```

Format

A `data.frame` object.

dataEnrichmentRatesStratified *Stratified Enrichment Dataset of Rates*

Description

A dataset containing the sample sizes and events of two groups. Use `getDataset(dataEnrichmentRatesStratified)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentRatesStratified
```

Format

A `data.frame` object.

dataEnrichmentSurvival *Enrichment Dataset of Survival Data*

Description

A dataset containing the log-rank statistics, events, and allocation ratios of two groups. Use `getDataset(dataEnrichmer` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentSurvival
```

Format

A `data.frame` object.

dataEnrichmentSurvivalStratified *Stratified Enrichment Dataset of Survival Data*

Description

A dataset containing the log-rank statistics, events, and allocation ratios of two groups. Use `getDataset(dataEnrichmer` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataEnrichmentSurvivalStratified
```

Format

A `data.frame` object.

dataMeans *One-Arm Dataset of Means*

Description

A dataset containing the sample sizes, means, and standard deviations of one group. Use `getDataset(dataMeans)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataMeans
```

Format

A `data.frame` object.

dataMultiArmMeans *Multi-Arm Dataset of Means*

Description

A dataset containing the sample sizes, means, and standard deviations of four groups. Use `getDataset(dataMultiArmMe` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataMultiArmMeans
```

Format

A `data.frame` object.

dataMultiArmRates *Multi-Arm Dataset of Rates*

Description

A dataset containing the sample sizes and events of three groups. Use `getDataset(dataMultiArmRates)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataMultiArmRates
```

Format

A `data.frame` object.

dataMultiArmSurvival *Multi-Arm Dataset of Survival Data*

Description

A dataset containing the log-rank statistics, events, and allocation ratios of three groups. Use `getDataset(dataMultiArmSurvival)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataMultiArmSurvival
```

Format

A `data.frame` object.

dataRates *One-Arm Dataset of Rates*

Description

A dataset containing the sample sizes and events of one group. Use `getDataset(dataRates)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataRates
```

Format

A `data.frame` object.

Dataset *Dataset*

Description

Basic class for datasets.

Details

`Dataset` is the basic class for

- `DatasetMeans`,
- `DatasetRates`,
- `DatasetSurvival`, and
- `DatasetEnrichmentSurvival`.

This basic class contains the fields `stages` and `groups` and several commonly used functions.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`groups` The group numbers. Is a numeric vector.

DatasetMeans *Dataset of Means*

Description

Class for a dataset of means.

Details

This object cannot be created directly; better use `getDataset` with suitable arguments to create a dataset of means.

Fields

`groups` The group numbers. Is a numeric vector.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`means` The means. Is a numeric vector of length number of stages times number of groups.

`stDevs` The standard deviations. Is a numeric vector of length number of stages times number of groups.

`overallSampleSizes` The overall, i.e., cumulative sample sizes. Is a numeric vector of length number of stages times number of groups.

`overallMeans` The overall, i.e., cumulative means. Is a numeric vector of length number of stages times number of groups.

`overallStDevs` The overall, i.e., cumulative standard deviations. Is a numeric vector of length number of stages times number of groups.

DatasetRates *Dataset of Rates*

Description

Class for a dataset of rates.

Details

This object cannot be created directly; better use `getDataset` with suitable arguments to create a dataset of rates.

Fields

`groups` The group numbers. Is a numeric vector.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`overallSampleSizes` The overall, i.e., cumulative sample sizes. Is a numeric vector of length number of stages times number of groups.

`events` The number of events in each group at each stage. Is a numeric vector of length number of stages times number of groups.

`overallEvents` The overall, i.e., cumulative events. Is a numeric vector of length number of stages times number of groups containing whole numbers.

DatasetSurvival *Dataset of Survival Data*

Description

Class for a dataset of survival data.

Details

This object cannot be created directly; better use `getDataset` with suitable arguments to create a dataset of survival data.

Fields

`groups` The group numbers. Is a numeric vector.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`events` The number of events in each group at each stage. Is a numeric vector of length number of stages times number of groups.

`overallEvents` The overall, i.e., cumulative events. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`allocationRatios` The observed allocation ratios. Is a numeric vector of length number of stages times number of groups.

`overallAllocationRatios` The cumulative allocation ratios. Is a numeric vector of length number of stages times number of groups.

`logRanks` The logrank test statistics at each stage of the trial. Is a numeric vector of length number of stages times number of groups.

`overallLogRanks` The overall, i.e., cumulative logrank test statistics. Is a numeric vector of length number of stages times number of groups.

---

dataSurvival *One-Arm Dataset of Survival Data*

Description

A dataset containing the log-rank statistics, events, and allocation ratios of one group. Use `getDataset(dataSurvival)` to create a dataset object that can be processed by `getAnalysisResults()`.

Usage

```
dataSurvival
```

Format

A `data.frame` object.

---

disableStartupMessages *Disable Startup Messages*

Description

This function disables the startup messages for the `rpact` package by setting the `rpact.startup.message.enabled` option to `FALSE`.

Usage

```
disableStartupMessages()
```

Details

Once this function is called, the startup messages will remain disabled until explicitly re-enabled using the `enableStartupMessages()` function. The current state is saved using the `saveOptions()` function.

Value

This function does not return a value. It is called for its side effects.

Examples

```
## Not run:
disableStartupMessages()

## End(Not run)
```

`enableStartupMessages` *Enable Startup Messages*

Description

This function enables the startup messages for the `rpact` package by setting the `rpact.startup.message.enabled` option to `TRUE`.

Usage

```
enableStartupMessages()
```

Details

Once this function is called, the startup messages will remain enabled until explicitly disabled using the `disableStartupMessages()` function. The current state is saved using the `saveOptions()` function.

Value

This function does not return a value. It is called for its side effects.

Examples

```
## Not run:
enableStartupMessages()

## End(Not run)
```

EventProbabilities *Event Probabilities*

Description

Class for the definition of event probabilities.

Details

EventProbabilities is a class for the definition of event probabilities.

Fields

time The time values. Is a numeric vector.

accrualTime The assumed accrual time intervals for the study. Is a numeric vector.

accrualIntensity The absolute accrual intensities. Is a numeric vector of length kMax.

kappa The shape of the Weibull distribution if kappa!=1. Is a numeric vector of length 1.

piecewiseSurvivalTime The time intervals for the piecewise definition of the exponential survival time cumulative distribution function. Is a numeric vector.

lambda1 The assumed hazard rate in the treatment group. Is a numeric vector of length kMax.

lambda2 The assumed hazard rate in the reference group. Is a numeric vector of length 1.

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

hazardRatio The hazard ratios under consideration. Is a numeric vector of length kMax.

dropoutRate1 The assumed drop-out rate in the treatment group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutRate2 The assumed drop-out rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutTime The assumed time for drop-out rates in the control and treatment group. Is a numeric vector of length 1.

maxNumberOfSubjects The maximum number of subjects for power calculations. Is a numeric vector.

overallEventProbabilities Deprecated field which will be removed in one of the next releases. Use cumulativeEventProbabilities instead.

cumulativeEventProbabilities The cumulative event probabilities in survival designs. Is a numeric vector.

eventProbabilities1 The event probabilities in treatment group 1. Is a numeric vector.

eventProbabilities2 The event probabilities in treatment group 2. Is a numeric vector.

FieldSet *Field Set*

Description

Basic class for field sets.

Details

The field set implements basic functions for a set of fields.

getAccrualTime *Get Accrual Time*

Description

Returns an AccrualTime object that contains the accrual time and the accrual intensity.

Usage

```
getAccrualTime(
  accrualTime = NA_real_,
  ...,
  accrualIntensity = NA_real_,
  accrualIntensityType = c("auto", "absolute", "relative"),
  maxNumberOfSubjects = NA_real_
)
```

Arguments

accrualTime	The assumed accrual time intervals for the study, default is c(0, 12) (for details see getAccrualTime()).
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
accrualIntensity	A numeric vector of accrual intensities, default is the relative intensity 0.1 (for details see getAccrualTime()).
accrualIntensityType	A character value specifying the accrual intensity input type. Must be one of "auto", "absolute", or "relative"; default is "auto", i.e., if all values are < 1 the type is "relative", otherwise it is "absolute".
maxNumberOfSubjects	The maximum number of subjects.

Value

Returns an AccrualTime object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Staggered patient entry

accrualTime is the time period of subjects' accrual in a study. It can be a value that defines the end of accrual or a vector. In this case, accrualTime can be used to define a non-constant accrual over time. For this, accrualTime is a vector that defines the accrual intervals. The first element of accrualTime must be equal to 0 and, additionally, accrualIntensity needs to be specified. accrualIntensity itself is a value or a vector (depending on the length of accrualTime) that defines the intensity how subjects enter the trial in the intervals defined through accrualTime.

accrualTime can also be a list that combines the definition of the accrual time and accrual intensity (see below and examples for details).

If the length of accrualTime and the length of accrualIntensity are the same (i.e., the end of accrual is undefined), maxNumberOfSubjects > 0 needs to be specified and the end of accrual is calculated. In that case, accrualIntensity is the number of subjects per time unit, i.e., the absolute accrual intensity.

If the length of accrualTime equals the length of accrualIntensity - 1 (i.e., the end of accrual is defined), maxNumberOfSubjects is calculated if the absolute accrual intensity is given. If all elements in accrualIntensity are smaller than 1, accrualIntensity defines the *relative* intensity how subjects enter the trial. For example, accrualIntensity = c(0.1, 0.2) specifies that in the second accrual interval the intensity is doubled as compared to the first accrual interval. The actual (absolute) accrual intensity is calculated for the calculated or given maxNumberOfSubjects. Note that the default is accrualIntensity = 0.1 meaning that the *absolute* accrual intensity will be calculated.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

getNumberOfSubjects() for calculating the number of subjects at given time points.

Examples

```
## Not run:
# Assume that in a trial the accrual after the first 6 months is doubled
# and the total accrual time is 30 months.
```

```
# Further assume that a total of 1000 subjects are entered in the trial.
# The number of subjects to be accrued in the first 6 months and afterwards
# is achieved through
getAccrualTime(
    accrualTime = c(0, 6, 30),
    accrualIntensity = c(0.1, 0.2), maxNumberOfSubjects = 1000
)

# The same result is obtained via the list based definition
getAccrualTime(
    list(
        "0 - <6"   = 0.1,
        "6 - <=30" = 0.2
    ),
    maxNumberOfSubjects = 1000
)

# Calculate the end of accrual at given absolute intensity:
getAccrualTime(
    accrualTime = c(0, 6),
    accrualIntensity = c(18, 36), maxNumberOfSubjects = 1000
)

# Via the list based definition this is
getAccrualTime(
    list(
        "0 - <6" = 18,
        ">=6" = 36
    ),
    maxNumberOfSubjects = 1000
)

# You can use an accrual time object in getSampleSizeSurvival() or
# getPowerSurvival().
# For example, if the maximum number of subjects and the follow up
# time needs to be calculated for a given effect size:
accrualTime <- getAccrualTime(
    accrualTime = c(0, 6, 30),
    accrualIntensity = c(0.1, 0.2)
)
getSampleSizeSurvival(accrualTime = accrualTime, pi1 = 0.4, pi2 = 0.2)

# Or if the power and follow up time needs to be calculated for given
# number of events and subjects:
accrualTime <- getAccrualTime(
    accrualTime = c(0, 6, 30),
    accrualIntensity = c(0.1, 0.2), maxNumberOfSubjects = 110
)
getPowerSurvival(
    accrualTime = accrualTime, pi1 = 0.4, pi2 = 0.2,
    maxNumberOfEvents = 46
)

# How to show accrual time details

# You can use a sample size or power object as argument for the function
# getAccrualTime():
```

```
sampleSize <- getSampleSizeSurvival(
    accrualTime = c(0, 6), accrualIntensity = c(22, 53),
    lambda2 = 0.05, hazardRatio = 0.8, followUpTime = 6
)
sampleSize
accrualTime <- getAccrualTime(sampleSize)
accrualTime

## End(Not run)
```

`getAnalysisResults` *Get Analysis Results*

Description

Calculates and returns the analysis results for the specified design and data.

Usage

```
getAnalysisResults(
  design,
  dataInput,
  ...,
  directionUpper = NA,
  thetaH0 = NA_real_,
  nPlanned = NA_real_,
  allocationRatioPlanned = 1,
  stage = NA_integer_,
  maxInformation = NULL,
  informationEpsilon = NULL
)
```

Arguments

`design`	The trial design.
`dataInput`	The summary data used for calculating the test results. This is either an element of `DatasetMeans`, of `DatasetRates`, or of `DatasetSurvival` and should be created with the function `getDataset()`. For more information see `getDataset()`.
`...`	Further arguments to be passed to methods (cf., separate functions in "See Also" below), e.g.,

`thetaH1` **and** `stDevH1` **(or** `assumedStDev` **/** `assumedStDevs`**),** `pi1`**,** `pi2`**, or** `piTreatments`**,** `piContr`

The assumed effect size, standard deviation or rates to calculate the conditional power if `nPlanned` is specified. For survival designs, `thetaH1` refers to the hazard ratio. For one-armed trials with binary outcome, only `pi1` can be specified, for two-armed trials with binary outcome, `pi1` and `pi2` can be specified referring to the assumed treatment and control rate, respectively. In multi-armed or enrichment designs, you can specify a value or a vector with elements referring to the treatment arms or the subpopulations, respectively. For testing rates, the parameters to be specified are `piTreatments` and `piControl` (multi-arm designs) and `piTreatments`

and piControls (enrichment designs).
If not specified, the conditional power is calculated under the assumption of observed effect sizes, standard deviations, rates, or hazard ratios.

iterations Iterations for simulating the power for Fisher's combination test. If the power for more than one remaining stages is to be determined for Fisher's combination test, it is estimated via simulation with specified iterations, the default is 1000.

seed Seed for simulating the conditional power for Fisher's combination test. See above, default is a random seed.

normalApproximation The type of computation of the p-values. Default is FALSE for testing means (i.e., the t test is used) and TRUE for testing rates and the hazard ratio. For testing rates, if normalApproximation = FALSE is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting, normalApproximation = FALSE has no effect.

equalVariances The type of t test. For testing means in two treatment groups, either the t test assuming that the variances are equal or the t test without assuming this, i.e., the test of Welch-Satterthwaite is calculated, default is TRUE.

stdErrorEstimate Estimate of standard error for calculation of final confidence intervals for comparing rates in two treatment groups, default is "pooled".

intersectionTest Defines the multiple test for the intersection hypotheses in the closed system of hypotheses when testing multiple hypotheses. Five options are available in multi-arm designs: "Dunnett", "Bonferroni", "Simes", "Sidak", and "Hierarchical", default is "Dunnett". Four options are available in population enrichment designs: "SpiessensDebois" (one subset only), "Bonferroni", "Simes", and "Sidak", default is "Simes".

varianceOption Defines the way to calculate the variance in multiple treatment arms (> 2) or population enrichment designs for testing means. For multiple arms, three options are available: "overallPooled", "pairwisePooled", and "notPooled", default is "overallPooled". For enrichment designs, the options are: "pooled", "pooledFromFull" (one subset only), and "notPooled", default is "pooled".

stratifiedAnalysis For enrichment designs, typically a stratified analysis should be chosen. For testing means and rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is TRUE.

directionUpper Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.

thetaH0 The null hypothesis value, default is 0 for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio).

For non-inferiority designs, thetaH0 is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value != 0 (or a value != 1 for testing the mean ratio) can be specified.
- *rates*: a value != 0 (or a value != 1 for testing the risk ratio pi1 / pi2) can be specified.

- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`nPlanned` The additional (i.e., "new" and not cumulative) sample size planned for each of the subsequent stages. The argument must be a vector with length equal to the number of remaining stages and contain the combined sample size from both treatment groups if two groups are considered. For survival outcomes, it should contain the planned number of additional events. For multi-arm designs, it is the per-comparison (combined) sample size. For enrichment designs, it is the (combined) sample size for the considered sub-population.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

`stage` The stage number (optional). Default: total number of existing stages in the data input.

`maxInformation` Positive value specifying the maximum information.

`informationEpsilon`
Positive integer value specifying the absolute information epsilon, which defines the maximum distance from the observed information to the maximum information that causes the final analysis. Updates at the final analysis in case the observed information at the final analysis is smaller ("under-running") than the planned maximum information `maxInformation`, default is 0. Alternatively, a floating-point number > 0 and < 1 can be specified to define a relative information epsilon.

Details

Given a design and a dataset, at given stage the function calculates the test results (effect sizes, stage-wise test statistics and p-values, overall p-values and test statistics, conditional rejection probability (CRP), conditional power, Repeated Confidence Intervals (RCIs), repeated overall p-values, and final stage p-values, median unbiased effect estimates, and final confidence intervals.

For designs with more than two treatments arms (multi-arm designs) or enrichment designs a closed combination test is performed. That is, additionally the statistics to be used in a closed testing procedure are provided.

The conditional power is calculated if the planned sample size for the subsequent stages (`nPlanned`) is specified. The conditional power is calculated either under the assumption of the observed effect or under the assumption of an assumed effect, that has to be specified (see above).
For testing rates in a two-armed trial, pi1 and pi2 typically refer to the rates in the treatment and the control group, respectively. This is not mandatory, however, and so pi1 and pi2 can be interchanged. In many-to-one multi-armed trials, piTreatments and piControl refer to the rates in the treatment arms and the one control arm, and so they cannot be interchanged. piTreatments and piControls in

enrichment designs can principally be interchanged, but we use the plural form to indicate that the rates can be differently specified for the sub-populations.

Median unbiased effect estimates and confidence intervals are calculated if a group sequential design or an inverse normal combination test design was chosen, i.e., it is not applicable for Fisher's p-value combination test design. For the inverse normal combination test design with more than two stages, a warning informs that the validity of the confidence interval is theoretically shown only if no sample size change was performed.

A final stage p-value for Fisher's combination test is calculated only if a two-stage design was chosen. For Fisher's combination test, the conditional power for more than one remaining stages is estimated via simulation.

Final stage p-values, median unbiased effect estimates, and final confidence intervals are not calculated for multi-arm and enrichment designs.

Value

Returns an `AnalysisResults` object. The following generics (R generic functions) are available for this result object:

- `names` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

See Also

`getObservedInformationRates()`

Other analysis functions: `getClosedCombinationTestResults()`, `getClosedConditionalDunnettTestResults()`, `getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`, `getFinalPValue()`, `getRepeatedConfidenceIntervals()`, `getRepeatedPValues()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
# Example 1 One-Sample t Test
# Perform an analysis within a three-stage group sequential design with
# O'Brien & Fleming boundaries and one-sample data with a continuous outcome
# where H0: mu = 1.2 is to be tested
dsnGS <- getDesignGroupSequential()
dataMeans <- getDataset(
    n = c(30, 30),
    means = c(1.96, 1.76),
```

```
    stDevs = c(1.92, 2.01)
)
getAnalysisResults(design = dsnGS, dataInput = dataMeans, thetaH0 = 1.2)

# You can obtain the results when performing an inverse normal combination test
# with these data by using the commands
dsnIN <- getDesignInverseNormal()
getAnalysisResults(design = dsnIN, dataInput = dataMeans, thetaH0 = 1.2)

# Example 2 Use Function Approach with Time to Event Data
# Perform an analysis within a use function approach according to an
# O'Brien & Fleming type use function and survival data where
# where H0: hazard ratio = 1 is to be tested. The events were observed
# over time and maxInformation = 120, informationEpsilon = 5 specifies
# that 116 > 120 - 5 observed events defines the final analysis.
design <- getDesignGroupSequential(typeOfDesign = "asOF")
dataSurvival <- getDataset(
    cumulativeEvents = c(33, 72, 116),
    cumulativeLogRanks = c(1.33, 1.88, 1.902)
)
getAnalysisResults(design,
    dataInput = dataSurvival,
    maxInformation = 120, informationEpsilon = 5
)

# Example 3 Multi-Arm Design
# In a four-stage combination test design with O'Brien & Fleming boundaries
# at the first stage the second treatment arm was dropped. With the Bonferroni
# intersection test, the results together with the CRP, conditional power
# (assuming a total of 40 subjects for each comparison and effect sizes 0.5
# and 0.8 for treatment arm 1 and 3, respectively, and standard deviation 1.2),
# RCIs and p-values of a closed adaptive test procedure are
# obtained as follows with the given data (treatment arm 4 refers to the
# reference group; displayed with summary and plot commands):
data <- getDataset(
    n1 = c(22, 23),
    n2 = c(21, NA),
    n3 = c(20, 25),
    n4 = c(25, 27),
    means1 = c(1.63, 1.51),
    means2 = c(1.4, NA),
    means3 = c(0.91, 0.95),
    means4 = c(0.83, 0.75),
    stds1 = c(1.2, 1.4),
    stds2 = c(1.3, NA),
    stds3 = c(1.1, 1.14),
    stds4 = c(1.02, 1.18)
)
design <- getDesignInverseNormal(kMax = 4)
x <- getAnalysisResults(design,
    dataInput = data, intersectionTest = "Bonferroni",
    nPlanned = c(40, 40), thetaH1 = c(0.5, NA, 0.8), assumedStDevs = 1.2
)
summary(x)
if (require(ggplot2)) plot(x, thetaRange = c(0, 0.8))
design <- getDesignConditionalDunnett(secondStageConditioning = FALSE)
y <- getAnalysisResults(design,
```

```
    dataInput = data,
    nPlanned = 40, thetaH1 = c(0.5, NA, 0.8), assumedStDevs = 1.2, stage = 1
)
summary(y)
if (require(ggplot2)) plot(y, thetaRange = c(0, 0.4))

# Example 4 Enrichment Design
# Perform an two-stage enrichment design analysis with O'Brien & Fleming boundaries
# where one sub-population (S1) and a full population (F) are considered as primary
# analysis sets. At interim, S1 is selected for further analysis and the sample
# size is increased accordingly. With the Spiessens & Debois intersection test,
# the results of a closed adaptive test procedure together with the CRP, repeated
# RCIs and p-values are obtained as follows with the given data (displayed with
# summary and plot commands):
design <- getDesignInverseNormal(kMax = 2, typeOfDesign = "OF")
dataS1 <- getDataset(
    means1 = c(13.2, 12.8),
    means2 = c(11.1, 10.8),
    stDev1 = c(3.4, 3.3),
    stDev2 = c(2.9, 3.5),
    n1 = c(21, 42),
    n2 = c(19, 39)
)
dataNotS1 <- getDataset(
    means1 = c(11.8, NA),
    means2 = c(10.5, NA),
    stDev1 = c(3.6, NA),
    stDev2 = c(2.7, NA),
    n1 = c(15, NA),
    n2 = c(13, NA)
)
dataBoth <- getDataset(S1 = dataS1, R = dataNotS1)
x <- getAnalysisResults(design,
    dataInput = dataBoth,
    intersectionTest = "SpiessensDebois",
    varianceOption = "pooledFromFull",
    stratifiedAnalysis = TRUE
)
summary(x)
if (require(ggplot2)) plot(x, type = 2)

## End(Not run)
```

`getClosedCombinationTestResults` *Get Closed Combination Test Results*

Description

Calculates and returns the results from the closed combination test in multi-arm and population enrichment designs.

Usage

```
getClosedCombinationTestResults(stageResults)
```

Arguments

`stageResults` The results at given stage, obtained from `getStageResults()`.

Value

Returns a `ClosedCombinationTestResults` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedConditionalDunnettTestResults()`, `getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`, `getFinalPValue()`, `getRepeatedConfidenceIntervals()`, `getRepeatedPValues()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
# In a four-stage combination test design with O'Brien & Fleming boundaries
# at the first stage the second treatment arm was dropped. With the Bonferroni
# intersection test, the results of a closed adaptive test procedure are
# obtained as follows with the given data (treatment arm 4 refers to the
# reference group):
data <- getDataset(
    n1 = c(22, 23),
    n2 = c(21, NA),
    n3 = c(20, 25),
    n4 = c(25, 27),
    means1 = c(1.63, 1.51),
    means2 = c(1.4, NA),
    means3 = c(0.91, 0.95),
    means4 = c(0.83, 0.75),
    stds1 = c(1.2, 1.4),
    stds2 = c(1.3, NA),
    stds3 = c(1.1, 1.14),
    stds4 = c(1.02, 1.18)
)
```

```
design <- getDesignInverseNormal(kMax = 4)
stageResults <- getStageResults(design,
    dataInput = data,
    intersectionTest = "Bonferroni"
)
getClosedCombinationTestResults(stageResults)

## End(Not run)
```

`getClosedConditionalDunnettTestResults` *Get Closed Conditional Dunnett Test Results*

Description

Calculates and returns the results from the closed conditional Dunnett test.

Usage

```
getClosedConditionalDunnettTestResults(
  stageResults,
  ...,
  stage = stageResults$stage
)
```

Arguments

`stageResults`	The results at given stage, obtained from `getStageResults()`.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`stage`	The stage number (optional). Default: total number of existing stages in the data input.

Details

For performing the conditional Dunnett test the design must be defined through the function `getDesignConditionalDunn`
See Koenig et al. (2008) and Wassmer & Brannath (2025), chapter 11 for details of the test procedure.

Value

Returns a `ClosedCombinationTestResults` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other analysis functions: getAnalysisResults(), getClosedCombinationTestResults(), getConditionalPower(), getConditionalRejectionProbabilities(), getFinalConfidenceInterval(), getFinalPValue(), getRepeatedConfidenceIntervals(), getRepeatedPValues(), getStageResults(), getTestActions()

Examples

```
## Not run:
# In a two-stage design a conditional Dunnett test should be performed
# where the  unconditional second stage p-values should be used for the
# test decision.
# At the first stage the second treatment arm was dropped. The results of
# a closed conditionsal Dunnett test are obtained as follows with the given
# data (treatment arm 4 refers to the reference group):
data <- getDataset(
    n1 = c(22, 23),
    n2 = c(21, NA),
    n3 = c(20, 25),
    n4 = c(25, 27),
    means1 = c(1.63, 1.51),
    means2 = c(1.4, NA),
    means3 = c(0.91, 0.95),
    means4 = c(0.83, 0.75),
    stds1 = c(1.2, 1.4),
    stds2 = c(1.3, NA),
    stds3 = c(1.1, 1.14),
    stds4 = c(1.02, 1.18)
)

# For getting the results of the closed test procedure, use the following commands:
design <- getDesignConditionalDunnett(secondStageConditioning = FALSE)
stageResults <- getStageResults(design, dataInput = data)
getClosedConditionalDunnettTestResults(stageResults)

## End(Not run)
```

---

getConditionalPower *Get Conditional Power*

Description

Calculates and returns the conditional power.

Usage

```
getConditionalPower(stageResults, ..., nPlanned, allocationRatioPlanned = 1)
```

Arguments

stageResults	The results at given stage, obtained from getStageResults().
...	Further (optional) arguments to be passed:
	thetaH1 **and** stDevH1 **(or** assumedStDev **/** assumedStDevs**),** pi1**,** pi2**, or** piTreatments**,** piContr The assumed effect size, standard deviation or rates to calculate the conditional power if nPlanned is specified. For survival designs, thetaH1 refers to the hazard ratio. For one-armed trials with binary outcome, only pi1 can be specified, for two-armed trials with binary outcome, pi1 and pi2 can be specified referring to the assumed treatment and control rate, respectively. In multi-armed or enrichment designs, you can specify a value or a vector with elements referring to the treatment arms or the sub-populations, respectively. For testing rates, the parameters to be specified are piTreatments and piControl (multi-arm designs) and piTreatments and piControls (enrichment designs). If not specified, the conditional power is calculated under the assumption of observed effect sizes, standard deviations, rates, or hazard ratios.
	iterations Iterations for simulating the power for Fisher's combination test. If the power for more than one remaining stages is to be determined for Fisher's combination test, it is estimated via simulation with specified iterations, the default is 1000.
	seed Seed for simulating the conditional power for Fisher's combination test. See above, default is a random seed.
nPlanned	The additional (i.e., "new" and not cumulative) sample size planned for each of the subsequent stages. The argument must be a vector with length equal to the number of remaining stages and contain the combined sample size from both treatment groups if two groups are considered. For survival outcomes, it should contain the planned number of additional events. For multi-arm designs, it is the per-comparison (combined) sample size. For enrichment designs, it is the (combined) sample size for the considered sub-population.
allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

Details

The conditional power is calculated if the planned sample size for the subsequent stages is specified. For testing rates in a two-armed trial, pi1 and pi2 typically refer to the rates in the treatment and the control group, respectively. This is not mandatory, however, and so pi1 and pi2 can be interchanged. In many-to-one multi-armed trials, piTreatments and piControl refer to the rates in the treatment arms and the one control arm, and so they cannot be interchanged. piTreatments and piControls in

enrichment designs can principally be interchanged, but we use the plural form to indicate that the rates can be differently specified for the sub-populations.

For Fisher's combination test, the conditional power for more than one remaining stages is estimated via simulation.

Value

Returns a ConditionalPowerResults object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

plot.StageResults() or plot.AnalysisResults() for plotting the conditional power.

Other analysis functions: getAnalysisResults(), getClosedCombinationTestResults(), getClosedConditionalD
getConditionalRejectionProbabilities(), getFinalConfidenceInterval(), getFinalPValue(),
getRepeatedConfidenceIntervals(), getRepeatedPValues(), getStageResults(), getTestActions()

Examples

```
## Not run:
data <- getDataset(
    n1     = c(22, 13, 22, 13),
    n2     = c(22, 11, 22, 11),
    means1 = c(1, 1.1, 1, 1),
    means2 = c(1.4, 1.5, 1, 2.5),
    stds1  = c(1, 2, 2, 1.3),
    stds2  = c(1, 2, 2, 1.3)
)
stageResults <- getStageResults(
    getDesignGroupSequential(kMax = 4),
    dataInput = data, stage = 2, directionUpper = FALSE
)
getConditionalPower(stageResults, thetaH1 = -0.4,
    nPlanned = c(64, 64), assumedStDev = 1.5,
    allocationRatioPlanned = 3
)

## End(Not run)
```

`getConditionalRejectionProbabilities`
Get Conditional Rejection Probabilities

Description

Calculates the conditional rejection probabilities (CRP) for given test results.

Usage

```
getConditionalRejectionProbabilities(stageResults, ...)
```

Arguments

`stageResults`	The results at given stage, obtained from `getStageResults()`.
`...`	Further (optional) arguments to be passed:
	`iterations` Iterations for simulating the conditional rejection probabilities for Fisher's combination test. For checking purposes, it can be estimated via simulation with specified `iterations`.
	`seed` Seed for simulating the conditional rejection probabilities for Fisher's combination test. See above, default is a random seed.

Details

The conditional rejection probability is the probability, under H0, to reject H0 in one of the subsequent (remaining) stages. The probability is calculated using the specified design. For testing rates and the survival design, the normal approximation is used, i.e., it is calculated with the use of the prototype case testing a mean for normally distributed data with known variance.

The conditional rejection probabilities are provided up to the specified stage.

For Fisher's combination test, you can check the validity of the CRP calculation via simulation.

Value

Returns a `numeric` vector of length `kMax` or in case of multi-arm stage results a `matrix` (each column represents a stage, each row a comparison) containing the conditional rejection probabilities.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedCombinationTestResults()`, `getClosedConditionalD`
`getConditionalPower()`, `getFinalConfidenceInterval()`, `getFinalPValue()`, `getRepeatedConfidenceInterva`
`getRepeatedPValues()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
# Calculate CRP for a Fisher's combination test design with
# two remaining stages and check the results by simulation.
design <- getDesignFisher(
    kMax = 4, alpha = 0.01,
    informationRates = c(0.1, 0.3, 0.8, 1)
)
```

```
data <- getDataset(n = c(40, 40), events = c(20, 22))
sr <- getStageResults(design, data, thetaH0 = 0.4)
getConditionalRejectionProbabilities(sr)
getConditionalRejectionProbabilities(sr,
    simulateCRP = TRUE,
    seed = 12345, iterations = 10000
)

## End(Not run)
```

`getData` *Get Simulation Data*

Description

Returns the aggregated simulation data.

Usage

```
getData(x)

getData.SimulationResults(x)
```

Arguments

x	A SimulationResults object created by getSimulationMeans(), getSimulationRates(), getSimulationSurvival(), getSimulationMultiArmMeans(), getSimulationMultiArmRates(), or getSimulationMultiArmSurvival().

Details

This function can be used to get the aggregated simulated data from an simulation results object, for example, obtained by getSimulationSurvival(). In this case, the data frame contains the following columns:

1. iterationNumber: The number of the simulation iteration.
2. stageNumber: The stage.
3. pi1: The assumed or derived event rate in the treatment group.
4. pi2: The assumed or derived event rate in the control group.
5. hazardRatio: The hazard ratio under consideration (if available).
6. analysisTime: The analysis time.
7. numberOfSubjects: The number of subjects under consideration when the (interim) analysis takes place.
8. eventsPerStage1: The observed number of events per stage in treatment group 1.
9. eventsPerStage2: The observed number of events per stage in treatment group 2.
10. eventsPerStage: The observed number of events per stage in both treatment groups.
11. rejectPerStage: 1 if null hypothesis can be rejected, 0 otherwise.

12. `eventsNotAchieved`: 1 if number of events could not be reached with observed number of subjects, 0 otherwise.
13. `futilityPerStage`: 1 if study should be stopped for futility, 0 otherwise.
14. `testStatistic`: The test statistic that is used for the test decision, depends on which design was chosen (group sequential, inverse normal, or Fisher combination test)'
15. `logRankStatistic`: Z-score statistic which corresponds to a one-sided log-rank test at considered stage.
16. `conditionalPowerAchieved`: The conditional power for the subsequent stage of the trial for selected sample size and effect. The effect is either estimated from the data or can be user defined with `thetaH1` or `pi1H1` and `pi2H1`.
17. `trialStop`: `TRUE` if study should be stopped for efficacy or futility or final stage, `FALSE` otherwise.
18. `hazardRatioEstimateLR`: The estimated hazard ratio, derived from the log-rank statistic.

A subset of variables is provided for `getSimulationMeans()`, `getSimulationRates()`, `getSimulationMultiArmMeans`, `getSimulationMultiArmRates()`, or `getSimulationMultiArmSurvival()`.

Value

Returns a `data.frame`.

Examples

```
## Not run:
results <- getSimulationSurvival(
    pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, eventTime = 12,
    accrualTime = 24, plannedEvents = 40, maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)
data <- getData(results)
head(data)
dim(data)

## End(Not run)
```

`getDataset` *Get Dataset*

Description

Creates a dataset object and returns it.

Usage

```
getDataset(..., floatingPointNumbersEnabled = FALSE)

getDataSet(..., floatingPointNumbersEnabled = FALSE)
```

Arguments

...	A data.frame or some data vectors defining the dataset.
floatingPointNumbersEnabled	If TRUE, sample sizes and event numbers can be specified as floating-point numbers (this make sense, e.g., for theoretical comparisons); by default floatingPointNumbersEnabled = FALSE, i.e., samples sizes and event numbers defined as floating-point numbers will be truncated.

Details

The different dataset types DatasetMeans, of DatasetRates, or DatasetSurvival can be created as follows:

- An element of DatasetMeans for one sample is created by getDataset(sampleSizes =, means =, stDevs =) where sampleSizes, means, stDevs are vectors with stage-wise sample sizes, means and standard deviations of length given by the number of available stages.
- An element of DatasetMeans for two samples is created by getDataset(sampleSizes1 =, sampleSizes2 =, means1 =, means2 =, stDevs1 =, stDevs2 =) where sampleSizes1, sampleSizes2, means1, means2, stDevs1, stDevs2 are vectors with stage-wise sample sizes, means and standard deviations for the two treatment groups of length given by the number of available stages.
- An element of DatasetRates for one sample is created by getDataset(sampleSizes =, events =) where sampleSizes, events are vectors with stage-wise sample sizes and events of length given by the number of available stages.
- An element of DatasetRates for two samples is created by getDataset(sampleSizes1 =, sampleSizes2 =, events1 =, events2 =) where sampleSizes1, sampleSizes2, events1, events2 are vectors with stage-wise sample sizes and events for the two treatment groups of length given by the number of available stages.
- An element of DatasetSurvival is created by getDataset(events =, logRanks =, allocationRatios =) where events, logRanks, and allocation ratios are the stage-wise events, (one-sided) logrank statistics, and allocation ratios.
- An element of DatasetMeans, DatasetRates, and DatasetSurvival for more than one comparison is created by adding subsequent digits to the variable names. The system can analyze these data in a multi-arm many-to-one comparison setting where the group with the highest index represents the control group.

Prefix overall[Capital case of first letter of variable name]... for the variable names enables entering the overall (cumulative) results and calculates stage-wise statistics. Since rpact version 3.2, the prefix cumulative[Capital case of first letter of variable name]... or cum[Capital case of first letter of variable name]... can alternatively be used for this.

n can be used in place of samplesizes.

Note that in survival design usually the overall (cumulative) events and logrank test statistics are provided in the output, so getDataset(cumulativeEvents=, cumulativeLogRanks =, cumulativeAllocationRatios =) is the usual command for entering survival data. Note also that for cumulativeLogranks also the z scores from a Cox regression can be used.

For multi-arm designs, the index refers to the considered comparison. For example, getDataset(events1=c(13, 33), logRanks1 = c(1.23, 1.55), events2 = c(16, NA), logRanks2

= c(1.55, NA))
refers to the case where one active arm (1) is considered at both stages whereas active arm 2 was dropped at interim. Number of events and logrank statistics are entered for the corresponding comparison to control (see Examples).

For enrichment designs, the comparison of two samples is provided for an unstratified (sub-population wise) or stratified data input.
For non-stratified (sub-population wise) data input the data sets are defined for the sub-populations S1, S2, ..., F, where F refers to the full populations. Use of `getDataset(S1 = , S2, ..., F = )` defines the data set to be used in `getAnalysisResults()` (see examples)
For stratified data input the data sets are defined for the strata S1, S12, S2, ..., R, where R refers to the remainder of the strata such that the union of all sets is the full population. Use of `getDataset(S1 = , S12 = , S2, ..., R = )` defines the data set to be used in `getAnalysisResults()` (see examples)
For survival data, for enrichment designs the log-rank statistics can only be entered as stratified log-rank statistics in order to provide strong control of Type I error rate. For stratified data input, the variables to be specified in `getDataset()` are `cumEvents`, `cumExpectedEvents`, `cumVarianceEvents`, and `cumAllocationRatios` or `overallEvents`, `overallExpectedEvents`, `overallVarianceEvents`, and `overallAllocationRatios`. From this, (stratified) log-rank tests and and the independent increments are calculated.

Value

Returns a `Dataset` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

Examples

```
## Not run:
# Create a Dataset of Means (one group):
datasetOfMeans <- getDataset(
    n      = c(22, 11, 22, 11),
    means  = c(1, 1.1, 1, 1),
    stDevs = c(1, 2, 2, 1.3)
)
datasetOfMeans
datasetOfMeans$show(showType = 2)

datasetOfMeans2 <- getDataset(
    cumulativeSampleSizes = c(22, 33, 55, 66),
    cumulativeMeans = c(1.000, 1.033, 1.020, 1.017),
    cumulativeStDevs = c(1.00, 1.38, 1.64, 1.58)
)
datasetOfMeans2
datasetOfMeans2$show(showType = 2)
as.data.frame(datasetOfMeans2)

# Create a Dataset of Means (two groups):
```

```
datasetOfMeans3 <- getDataset(
    n1 = c(22, 11, 22, 11),
    n2 = c(22, 13, 22, 13),
    means1  = c(1, 1.1, 1, 1),
    means2  = c(1.4, 1.5, 3, 2.5),
    stDevs1 = c(1, 2, 2, 1.3),
    stDevs2 = c(1, 2, 2, 1.3)
)
datasetOfMeans3

datasetOfMeans4 <- getDataset(
    cumulativeSampleSizes1 = c(22, 33, 55, 66),
    cumulativeSampleSizes2 = c(22, 35, 57, 70),
    cumulativeMeans1  = c(1, 1.033, 1.020, 1.017),
    cumulativeMeans2  = c(1.4, 1.437, 2.040, 2.126),
    cumulativeStDevs1 = c(1, 1.38, 1.64, 1.58),
    cumulativeStDevs2 = c(1, 1.43, 1.82, 1.74)
)
datasetOfMeans4

df <- data.frame(
    stages = 1:4,
    n1      = c(22, 11, 22, 11),
    n2      = c(22, 13, 22, 13),
    means1  = c(1, 1.1, 1, 1),
    means2  = c(1.4, 1.5, 3, 2.5),
    stDevs1 = c(1, 2, 2, 1.3),
    stDevs2 = c(1, 2, 2, 1.3)
)
datasetOfMeans5 <- getDataset(df)
datasetOfMeans5

# Create a Dataset of Means (three groups) where the comparison of
# treatment arm 1 to control is dropped at the second interim stage:
datasetOfMeans6 <- getDataset(
   cumN1      = c(22, 33, NA),
   cumN2      = c(20, 34, 56),
   cumN3      = c(22, 31, 52),
   cumMeans1  = c(1.64, 1.54, NA),
   cumMeans2  = c(1.7, 1.5, 1.77),
   cumMeans3  = c(2.5, 2.06, 2.99),
   cumStDevs1 = c(1.5, 1.9, NA),
   cumStDevs2 = c(1.3, 1.3, 1.1),
   cumStDevs3 = c(1, 1.3, 1.8))
datasetOfMeans6

# Create a Dataset of Rates (one group):
datasetOfRates <- getDataset(
    n = c(8, 10, 9, 11),
    events = c(4, 5, 5, 6)
)
datasetOfRates

# Create a Dataset of Rates (two groups):
datasetOfRates2 <- getDataset(
    n2      = c(8, 10, 9, 11),
    n1      = c(11, 13, 12, 13),
```

```
    events2 = c(3, 5, 5, 6),
    events1 = c(10, 10, 12, 12)
)
datasetOfRates2

# Create a Dataset of Rates (three groups) where the comparison of
# treatment arm 2 to control is dropped at the first interim stage:
datasetOfRates3 <- getDataset(
    cumN1      = c(22, 33, 44),
    cumN2      = c(20, NA, NA),
    cumN3      = c(20, 34, 44),
    cumEvents1 = c(11, 14, 22),
    cumEvents2 = c(17, NA, NA),
    cumEvents3 = c(17, 19, 33))
datasetOfRates3

# Create a Survival Dataset
datasetSurvival <- getDataset(
    cumEvents = c(8, 15, 19, 31),
    cumAllocationRatios = c(1, 1, 1, 2),
    cumLogRanks = c(1.52, 1.98, 1.99, 2.11)
)
datasetSurvival

# Create a Survival Dataset with four comparisons where treatment
# arm 2 was dropped at the first interim stage, and treatment arm 4
# at the second.
datasetSurvival2 <- getDataset(
    cumEvents1   = c(18, 45, 56),
    cumEvents2   = c(22, NA, NA),
    cumEvents3   = c(12, 41, 56),
    cumEvents4   = c(27, 56, NA),
    cumLogRanks1 = c(1.52, 1.98, 1.99),
    cumLogRanks2 = c(3.43, NA, NA),
    cumLogRanks3 = c(1.45, 1.67, 1.87),
    cumLogRanks4 = c(1.12, 1.33, NA)
)
datasetSurvival2

# Enrichment: Stratified and unstratified data input
# The following data are from one study. Only the first
# (stratified) data input enables a stratified analysis.

# Stratified data input
S1 <- getDataset(
    sampleSize1 = c(18, 17),
    sampleSize2 = c(12, 33),
    mean1       = c(125.6, 111.1),
    mean2       = c(107.7, 77.7),
    stDev1      = c(120.1, 145.6),
    stDev2      = c(128.5, 133.3))
S2 <- getDataset(
    sampleSize1 = c(11, NA),
    sampleSize2 = c(14, NA),
    mean1       = c(100.1, NA),
    mean2      = c( 68.3, NA),
    stDev1      = c(116.8, NA),
```

```
    stDev2      = c(124.0, NA))
S12 <- getDataset(
    sampleSize1 = c(21, 17),
    sampleSize2 = c(21, 12),
    mean1       = c(135.9, 117.7),
    mean2       = c(84.9, 107.7),
    stDev1      = c(185.0, 92.3),
    stDev2      = c(139.5, 107.7))
R <- getDataset(
    sampleSize1 = c(19, NA),
    sampleSize2 = c(33, NA),
    mean1       = c(142.4, NA),
    mean2       = c(77.1, NA),
    stDev1      = c(120.6, NA),
    stDev2      = c(163.5, NA))
dataEnrichment <- getDataset(S1 = S1, S2 = S2, S12 = S12, R = R)
dataEnrichment

# Unstratified data input
S1N <- getDataset(
    sampleSize1 = c(39, 34),
    sampleSize2 = c(33, 45),
    stDev1      = c(156.503, 120.084),
    stDev2      = c(134.025, 126.502),
    mean1       = c(131.146, 114.4),
    mean2       = c(93.191, 85.7))
S2N <- getDataset(
    sampleSize1 = c(32, NA),
    sampleSize2 = c(35, NA),
    stDev1      = c(163.645, NA),
    stDev2      = c(131.888, NA),
    mean1       = c(123.594, NA),
    mean2       = c(78.26, NA))
F <- getDataset(
    sampleSize1 = c(69, NA),
    sampleSize2 = c(80, NA),
    stDev1      = c(165.468, NA),
    stDev2      = c(143.979, NA),
    mean1       = c(129.296, NA),
    mean2       = c(82.187, NA))
dataEnrichmentN <- getDataset(S1 = S1N, S2 = S2N, F = F)
dataEnrichmentN

## End(Not run)
```

`getDesignCharacteristics` *Get Design Characteristics*

Description

Calculates the characteristics of a design and returns it.

Usage

```
getDesignCharacteristics(design = NULL, ...)
```

Arguments

design	The trial design.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

Details

Calculates the inflation factor (IF), the expected reduction in sample size under H1, under H0, and under a value in between H0 and H1. Furthermore, absolute information values are calculated under the prototype case testing H0: mu = 0 against H1: mu = 1.

Value

Returns a TrialDesignCharacteristics object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other design functions: getDesignConditionalDunnett(), getDesignFisher(), getDesignGroupSequential(), getDesignInverseNormal(), getGroupSequentialProbabilities(), getPowerAndAverageSampleNumber()

Examples

```
## Not run:
# Calculate design characteristics for a three-stage O'Brien & Fleming
# design at power 90% and compare it with Pocock's design.
getDesignCharacteristics(getDesignGroupSequential(beta = 0.1))
getDesignCharacteristics(getDesignGroupSequential(beta = 0.1, typeOfDesign = "P"))

## End(Not run)
```

`getDesignConditionalDunnett`

Get Design Conditional Dunnett Test

Description

Defines the design to perform an analysis with the conditional Dunnett test.

Usage

```
getDesignConditionalDunnett(
  alpha = 0.025,
  informationAtInterim = 0.5,
  ...,
  secondStageConditioning = TRUE,
  directionUpper = NA
)
```

Arguments

`alpha`	The significance level alpha, default is `0.025`. Must be a positive numeric of length 1.
`informationAtInterim`	The information to be expected at interim, default is `informationAtInterim = 0.5`.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`secondStageConditioning`	The way the second stage p-values are calculated within the closed system of hypotheses. If `secondStageConditioning = FALSE` is specified, the unconditional adjusted p-values are used, otherwise conditional adjusted p-values are calculated, default is `secondStageConditioning = TRUE` (for details, see Koenig et al., 2008).
`directionUpper`	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

Details

For performing the conditional Dunnett test the design must be defined through this function. You can define the information fraction and the way of how to compute the second stage p-values only in the design definition, and not in the analysis call.
See `getClosedConditionalDunnettTestResults()` for an example and Koenig et al. (2008) and Wassmer & Brannath (2025), chapter 11 for details of the test procedure.

Value

Returns a `TrialDesign` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,

- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other design functions: getDesignCharacteristics(), getDesignFisher(), getDesignGroupSequential(), getDesignInverseNormal(), getGroupSequentialProbabilities(), getPowerAndAverageSampleNumber()

---

getDesignFisher *Get Design Fisher*

Description

Performs Fisher's combination test and returns critical values for this design.

Usage

```
getDesignFisher(
  ...,
  kMax = NA_integer_,
  alpha = NA_real_,
  method = c("equalAlpha", "fullAlpha", "noInteraction", "userDefinedAlpha"),
  userAlphaSpending = NA_real_,
  alpha0Vec = NA_real_,
  alpha0Scale = c("pValue", "zValue", "condPowerAtObserved", "predictivePower"),
  informationRates = NA_real_,
  sided = 1,
  bindingFutility = NA,
  directionUpper = NA,
  tolerance = 1e-14,
  iterations = 0,
  seed = NA_real_
)
```

Arguments

...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
kMax	The maximum number of stages K. Must be a positive integer of length 1 (default value is 3). The maximum selectable kMax is 20 for group sequential or inverse normal and 6 for Fisher combination test designs.
alpha	The significance level alpha, default is 0.025. Must be a positive numeric of length 1.
method	"equalAlpha", "fullAlpha", "noInteraction", or "userDefinedAlpha", default is "equalAlpha" (for details, see Wassmer, 1999).
userAlphaSpending	The user defined alpha spending. Numeric vector of length kMax containing the cumulative alpha-spending (Type I error rate) up to each interim stage: 0 <= alpha_1 <= ... <= alpha_K <= alpha.
alpha0Vec	Stopping for futility bounds for stage-wise p-values.
alpha0Scale	Character. The scale of the futility bounds. Must be one of "pValue", "zValue", "condPowerAtObserved", or "predictivePower". Default is "pValue".
informationRates	The information rates t_1, ..., t_kMax (that must be fixed prior to the trial), default is (1:kMax) / kMax. For the weighted inverse normal design, the weights are derived through w_1 = sqrt(t_1), and w_k = sqrt(t_k - t_(k-1)). For the weighted Fisher's combination test, the weights (scales) are w_k = sqrt((t_k - t_(k-1)) / t_1) (see the documentation).
sided	Is the alternative one-sided (1) or two-sided (2), default is 1. Must be a positive integer of length 1.
bindingFutility	If bindingFutility = TRUE is specified the calculation of the critical values is affected by the futility bounds (default is TRUE).
directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.
tolerance	The numerical tolerance, default is 1e-14.
iterations	The number of simulation iterations, e.g., getDesignFisher(iterations = 100000) checks the validity of the critical values for the design. The default value of iterations is 0, i.e., no simulation will be executed.
seed	Seed for simulating the power for Fisher's combination test. See above, default is a random seed.

Details

getDesignFisher() calculates the critical values and stage levels for Fisher's combination test as described in Bauer (1989), Bauer and Koehne (1994), Bauer and Roehmel (1995), and Wassmer (1999) for equally and unequally sized stages.

Value

Returns a TrialDesign object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

getDesignSet() for creating a set of designs to compare.

getFutilityBounds() for the specification of futility bounds on scales other than the p-value scale.

Vignette: Enhanced Futility Bounds Specification

Other design functions: getDesignCharacteristics(), getDesignConditionalDunnett(), getDesignGroupSequen
getDesignInverseNormal(), getGroupSequentialProbabilities(), getPowerAndAverageSampleNumber()

Examples

```
## Not run:
# Calculate critical values for a two-stage Fisher's combination test
# with full level alpha = 0.05 at the final stage and stopping for
# futility bound alpha0 = 0.50, as described in Bauer and Koehne (1994).
getDesignFisher(kMax = 2, method = "fullAlpha", alpha = 0.05, alpha0Vec = 0.50)

## End(Not run)
```

---

getDesignGroupSequential

Get Design Group Sequential

Description

Provides adjusted boundaries and defines a group sequential design.

Usage

```
getDesignGroupSequential(
  ...,
  kMax = NA_integer_,
  alpha = NA_real_,
  beta = NA_real_,
  sided = 1L,
```

```
  informationRates = NA_real_,
  futilityBounds = NA_real_,
 typeOfDesign = c("OF", "P", "WT", "PT", "HP", "WToptimum", "asP", "asOF", "asKD",
    "asHSD", "asUser", "noEarlyEfficacy"),
  deltaWT = NA_real_,
  deltaPT1 = NA_real_,
  deltaPT0 = NA_real_,
  optimizationCriterion = c("ASNH1", "ASNIFH1", "ASNsum"),
  gammaA = NA_real_,
  typeBetaSpending = c("none", "bsP", "bsOF", "bsKD", "bsHSD", "bsUser"),
  userAlphaSpending = NA_real_,
  userBetaSpending = NA_real_,
  efficacyStops = NA,
  futilityStops = NA,
  gammaB = NA_real_,
  bindingFutility = NA,
 futilityBoundsScale = c("zValue", "pValue", "reverseCondPower", "condPowerAtObserved",
    "predictivePower"),
  directionUpper = NA,
  betaAdjustment = NA,
  constantBoundsHP = 3,
  twoSidedPower = NA,
  delayedInformation = NA_real_,
  tolerance = 1e-08
)
```

Arguments

`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`kMax`	The maximum number of stages K. Must be a positive integer of length 1 (default value is 3). The maximum selectable `kMax` is `20` for group sequential or inverse normal and 6 for Fisher combination test designs.
`alpha`	The significance level alpha, default is `0.025`. Must be a positive numeric of length 1.
`beta`	Type II error rate, necessary for providing sample size calculations (e.g., `getSampleSizeMeans()`), beta spending function designs, or optimum designs, default is `0.20`. Must be a positive numeric of length 1.
`sided`	Is the alternative one-sided (1) or two-sided (2), default is 1. Must be a positive integer of length 1.
`informationRates`	The information rates t_1, ..., t_kMax (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. For the weighted inverse normal design, the weights are derived through w_1 = sqrt(t_1), and w_k = sqrt(t_k - t_(k-1)). For the weighted Fisher's combination test, the weights (scales) are w_k = sqrt((t_k - t_(k-1)) / t_1) (see the documentation).
`futilityBounds`	The futility bounds, defined on the scale defined by `futilityBoundsScale`. (numeric vector of length `kMax - 1`).
`typeOfDesign`	The type of design. Type of design is one of the following: O'Brien & Fleming (`"OF"`), Pocock (`"P"`), Wang & Tsiatis Delta class (`"WT"`), Pampallona & Tsiatis (`"PT"`), Haybittle & Peto ("HP"), Optimum design within Wang & Tsiatis class

("WToptimum"), O'Brien & Fleming type alpha spending ("asOF"), Pocock type alpha spending ("asP"), Kim & DeMets alpha spending ("asKD"), Hwang, Shi & DeCani alpha spending ("asHSD"), user defined alpha spending ("asUser"), no early efficacy stop ("noEarlyEfficacy"), default is "OF".

deltaWT — Delta for Wang & Tsiatis Delta class.

deltaPT1 — Delta1 for Pampallona & Tsiatis class rejecting H0 boundaries.

deltaPT0 — Delta0 for Pampallona & Tsiatis class rejecting H1 boundaries.

optimizationCriterion — Optimization criterion for optimum design within Wang & Tsiatis class ("ASNH1", "ASNIFH1", "ASNsum"), default is "ASNH1", see details.

gammaA — Parameter for alpha spending function.

typeBetaSpending — Type of beta spending. Type of of beta spending is one of the following: O'Brien & Fleming type beta spending, Pocock type beta spending, Kim & DeMets beta spending, Hwang, Shi & DeCani beta spending, user defined beta spending ("bsOF", "bsP", "bsKD", "bsHSD", "bsUser", default is "none").

userAlphaSpending — The user defined alpha spending. Numeric vector of length kMax containing the cumulative alpha-spending (Type I error rate) up to each interim stage: 0 <= alpha_1 <= ... <= alpha_K <= alpha.

userBetaSpending — The user defined beta spending. Vector of length kMax containing the cumulative beta-spending up to each interim stage.

efficacyStops — Logical vector of length kMax - 1 indicating efficacy stops. Default is NA.

futilityStops — Logical vector of length kMax - 1 indicating futility stops. Default is NA.

gammaB — Parameter for beta spending function.

bindingFutility — Logical. If bindingFutility = TRUE is specified the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is FALSE).

futilityBoundsScale — Character. The scale of the futility bounds. Must be one of "zValue", "pValue", "reverseCondPower", "condPowerAtObserved", or "predictivePower". Default is "zValue".

directionUpper — Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.

betaAdjustment — For two-sided beta spending designs, if betaAdjustement = TRUE a linear adjustment of the beta spending values is performed if an overlapping of decision regions for futility stopping at earlier stages occurs, otherwise no adjustment is performed (default is TRUE).

constantBoundsHP — The constant bounds up to stage kMax - 1 for the Haybittle & Peto design (default is 3).

twoSidedPower — For two-sided testing, if twoSidedPower = TRUE is specified the sample size calculation is performed by considering both tails of the distribution. Default is FALSE, i.e., it is assumed that one tail probability is equal to 0 or the power should be directed to one part.

delayedInformation
: Delay of information for delayed response designs. Can be a numeric value or a numeric vector of length kMax - 1

tolerance
: The numerical tolerance, default is 1e-08.

Details

Depending on typeOfDesign some parameters are specified, others not. For example, only if typeOfDesign "asHSD" is selected, gammaA needs to be specified.

If an alpha spending approach was specified ("asOF", "asP", "asKD", "asHSD", or "asUser") additionally a beta spending function can be specified to produce futility bounds.

For optimum designs, "ASNH1" minimizes the expected sample size under H1, "ASNIFH1" minimizes the sum of the maximum sample and the expected sample size under H1, and "ASNsum" minimizes the sum of the maximum sample size, the expected sample size under a value midway H0 and H1, and the expected sample size under H1.

Value

Returns a TrialDesign object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

getDesignSet() for creating a set of designs to compare different designs.

getFutilityBounds() for the specification of futility bounds on scales other than the z-value scale.

Vignette: Enhanced Futility Bounds Specification

Other design functions: getDesignCharacteristics(), getDesignConditionalDunnett(), getDesignFisher(), getDesignInverseNormal(), getGroupSequentialProbabilities(), getPowerAndAverageSampleNumber()

Examples

```
## Not run:
# Calculate two-sided critical values for a four-stage
# Wang & Tsiatis design with Delta = 0.25 at level alpha = 0.05
getDesignGroupSequential(kMax = 4, alpha = 0.05, sided = 2,
    typeOfDesign = "WT", deltaWT = 0.25)
```

```
# Calculate one-sided critical values and binding futility bounds for a three-stage
# design with alpha- and beta-spending functions according to Kim & DeMets with gamma = 2.5
# (planned informationRates as specified, default alpha = 0.025 and beta = 0.2)
getDesignGroupSequential(kMax = 3, informationRates = c(0.3, 0.75, 1),
    typeOfDesign = "asKD", gammaA = 2.5, typeBetaSpending = "bsKD",
    gammaB = 2.5, bindingFutility = TRUE)

# Calculate the Pocock type alpha spending critical values if the first
# interim analysis was performed after 40% of the maximum information was observed
# and the second after 70% of the maximum information was observed (default alpha = 0.025)
getDesignGroupSequential(informationRates = c(0.4, 0.7), typeOfDesign = "asP")

## End(Not run)
```

`getDesignInverseNormal` *Get Design Inverse Normal*

Description

Provides adjusted boundaries and defines a group sequential design for its use in the inverse normal combination test.

Usage

```
getDesignInverseNormal(
  ...,
  kMax = NA_integer_,
  alpha = NA_real_,
  beta = NA_real_,
  sided = 1L,
  informationRates = NA_real_,
  futilityBounds = NA_real_,
  typeOfDesign = c("OF", "P", "WT", "PT", "HP", "WToptimum", "asP", "asOF", "asKD",
    "asHSD", "asUser", "noEarlyEfficacy"),
  deltaWT = NA_real_,
  deltaPT1 = NA_real_,
  deltaPT0 = NA_real_,
  optimizationCriterion = c("ASNH1", "ASNIFH1", "ASNsum"),
  gammaA = NA_real_,
  typeBetaSpending = c("none", "bsP", "bsOF", "bsKD", "bsHSD", "bsUser"),
  userAlphaSpending = NA_real_,
  userBetaSpending = NA_real_,
  efficacyStops = NA,
  futilityStops = NA,
  gammaB = NA_real_,
  bindingFutility = NA,
  futilityBoundsScale = c("zValue", "pValue", "reverseCondPower", "condPowerAtObserved",
    "predictivePower"),
  directionUpper = NA,
```

```
  betaAdjustment = NA,
  constantBoundsHP = 3,
  twoSidedPower = NA,
  tolerance = 1e-08
)
```

Arguments

...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
kMax	The maximum number of stages K. Must be a positive integer of length 1 (default value is 3). The maximum selectable kMax is 20 for group sequential or inverse normal and 6 for Fisher combination test designs.
alpha	The significance level alpha, default is 0.025. Must be a positive numeric of length 1.
beta	Type II error rate, necessary for providing sample size calculations (e.g., getSampleSizeMeans()), beta spending function designs, or optimum designs, default is 0.20. Must be a positive numeric of length 1.
sided	Is the alternative one-sided (1) or two-sided (2), default is 1. Must be a positive integer of length 1.
informationRates	The information rates t_1, ..., t_kMax (that must be fixed prior to the trial), default is (1:kMax) / kMax. For the weighted inverse normal design, the weights are derived through w_1 = sqrt(t_1), and w_k = sqrt(t_k - t_(k-1)). For the weighted Fisher's combination test, the weights (scales) are w_k = sqrt((t_k - t_(k-1)) / t_1) (see the documentation).
futilityBounds	The futility bounds, defined on the scale defined by futilityBoundsScale. (numeric vector of length kMax - 1).
typeOfDesign	The type of design. Type of design is one of the following: O'Brien & Fleming ("OF"), Pocock ("P"), Wang & Tsiatis Delta class ("WT"), Pampallona & Tsiatis ("PT"), Haybittle & Peto ("HP"), Optimum design within Wang & Tsiatis class ("WToptimum"), O'Brien & Fleming type alpha spending ("asOF"), Pocock type alpha spending ("asP"), Kim & DeMets alpha spending ("asKD"), Hwang, Shi & DeCani alpha spending ("asHSD"), user defined alpha spending ("asUser"), no early efficacy stop ("noEarlyEfficacy"), default is "OF".
deltaWT	Delta for Wang & Tsiatis Delta class.
deltaPT1	Delta1 for Pampallona & Tsiatis class rejecting H0 boundaries.
deltaPT0	Delta0 for Pampallona & Tsiatis class rejecting H1 boundaries.
optimizationCriterion	Optimization criterion for optimum design within Wang & Tsiatis class ("ASNH1", "ASNIFH1", "ASNsum"), default is "ASNH1", see details.
gammaA	Parameter for alpha spending function.
typeBetaSpending	Type of beta spending. Type of of beta spending is one of the following: O'Brien & Fleming type beta spending, Pocock type beta spending, Kim & DeMets beta spending, Hwang, Shi & DeCani beta spending, user defined beta spending ("bsOF", "bsP", "bsKD", "bsHSD", "bsUser", default is "none").

`userAlphaSpending`
: The user defined alpha spending. Numeric vector of length `kMax` containing the cumulative alpha-spending (Type I error rate) up to each interim stage: `0 <= alpha_1 <= ... <= alpha_K <= alpha`.

`userBetaSpending`
: The user defined beta spending. Vector of length `kMax` containing the cumulative beta-spending up to each interim stage.

`efficacyStops`
: Logical vector of length `kMax - 1` indicating efficacy stops. Default is `NA`.

`futilityStops`
: Logical vector of length `kMax - 1` indicating futility stops. Default is `NA`.

`gammaB`
: Parameter for beta spending function.

`bindingFutility`
: Logical. If `bindingFutility = TRUE` is specified the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`).

`futilityBoundsScale`
: Character. The scale of the futility bounds. Must be one of `"zValue"`, `"pValue"`, `"reverseCondPower"`, `"condPowerAtObserved"`, or `"predictivePower"`. Default is `"zValue"`.

`directionUpper`
: Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`betaAdjustment`
: For two-sided beta spending designs, if `betaAdjustement = TRUE` a linear adjustment of the beta spending values is performed if an overlapping of decision regions for futility stopping at earlier stages occurs, otherwise no adjustment is performed (default is `TRUE`).

`constantBoundsHP`
: The constant bounds up to stage `kMax - 1` for the Haybittle & Peto design (default is `3`).

`twoSidedPower`
: For two-sided testing, if `twoSidedPower = TRUE` is specified the sample size calculation is performed by considering both tails of the distribution. Default is `FALSE`, i.e., it is assumed that one tail probability is equal to 0 or the power should be directed to one part.

`tolerance`
: The numerical tolerance, default is `1e-08`.

Details

Depending on `typeOfDesign` some parameters are specified, others not. For example, only if `typeOfDesign` `"asHSD"` is selected, `gammaA` needs to be specified.

If an alpha spending approach was specified (`"asOF"`, `"asP"`, `"asKD"`, `"asHSD"`, or `"asUser"`) additionally a beta spending function can be specified to produce futility bounds.

For optimum designs, `"ASNH1"` minimizes the expected sample size under H1, `"ASNIFH1"` minimizes the sum of the maximum sample and the expected sample size under H1, and `"ASNsum"` minimizes the sum of the maximum sample size, the expected sample size under a value midway H0 and H1, and the expected sample size under H1.

Value

Returns a `TrialDesign` object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

getDesignSet() for creating a set of designs to compare different designs.

Other design functions: getDesignCharacteristics(), getDesignConditionalDunnett(), getDesignFisher(), getDesignGroupSequential(), getGroupSequentialProbabilities(), getPowerAndAverageSampleNumber()

Examples

```
## Not run:
# Calculate two-sided critical values for a four-stage
# Wang & Tsiatis design with Delta = 0.25 at level alpha = 0.05
getDesignInverseNormal(kMax = 4, alpha = 0.05, sided = 2,
    typeOfDesign = "WT", deltaWT = 0.25)

# Defines a two-stage design at one-sided alpha = 0.025 with provision of early stopping
# if the one-sided p-value exceeds 0.5 at interim and no early stopping for efficacy.
# The futility bound is non-binding.
getDesignInverseNormal(kMax = 2, typeOfDesign = "noEarlyEfficacy", futilityBounds = 0)

# Calculate one-sided critical values and binding futility bounds for a three-stage
# design with alpha- and beta-spending functions according to Kim & DeMets with gamma = 2.5
# (planned informationRates as specified, default alpha = 0.025 and beta = 0.2)
getDesignInverseNormal(kMax = 3, informationRates = c(0.3, 0.75, 1),
    typeOfDesign = "asKD", gammaA = 2.5, typeBetaSpending = "bsKD",
    gammaB = 2.5, bindingFutility = TRUE)

## End(Not run)
```

---

getDesignSet *Get Design Set*

Description

Creates a trial design set object and returns it.

Usage

```
getDesignSet(...)
```

Arguments

...	designs or design and one or more design parameters, e.g., deltaWT = c(0.1, 0.3, 0.4).

- design design

```
    sided = 2, typeOfDesign = "WT", deltaWT = 0.1
)
designSet <- getDesignSet(design = design, deltaWT = c(0.3, 0.4))
if (require(ggplot2)) plot(designSet, type = 1)

# Example 3 (use of designs instead of design)
d1 <- getDesignGroupSequential(
    alpha = 0.05, kMax = 2,
    sided = 1, beta = 0.2, typeOfDesign = "asHSD",
    gammaA = 0.5, typeBetaSpending = "bsHSD", gammaB = 0.5
)
d2 <- getDesignGroupSequential(
    alpha = 0.05, kMax = 4,
    sided = 1, beta = 0.2, typeOfDesign = "asP",
    typeBetaSpending = "bsP"
)
designSet <- getDesignSet(
    designs = c(d1, d2),
    variedParameters = c("typeOfDesign", "kMax")
)
if (require(ggplot2)) plot(designSet, type = 8, nMax = 20)

## End(Not run)
```

`getEventProbabilities` *Get Event Probabilities*

Description

Returns the event probabilities for specified parameters at given time vector.

Usage

```
getEventProbabilities(
  time,
  ...,
  accrualTime = c(0, 12),
  accrualIntensity = 0.1,
  accrualIntensityType = c("auto", "absolute", "relative"),
  kappa = 1,
  piecewiseSurvivalTime = NA_real_,
  lambda2 = NA_real_,
  lambda1 = NA_real_,
  allocationRatioPlanned = 1,
  hazardRatio = NA_real_,
  dropoutRate1 = 0,
  dropoutRate2 = 0,
  dropoutTime = 12,
  maxNumberOfSubjects = NA_real_
)
```

Arguments

time	A numeric vector with time values.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
accrualTime	The assumed accrual time intervals for the study, default is c(0, 12) (for details see getAccrualTime()).
accrualIntensity	A numeric vector of accrual intensities, default is the relative intensity 0.1 (for details see getAccrualTime()).
accrualIntensityType	A character value specifying the accrual intensity input type. Must be one of "auto", "absolute", or "relative"; default is "auto", i.e., if all values are < 1 the type is "relative", otherwise it is "absolute".
kappa	A numeric value > 0. A kappa != 1 will be used for the specification of the shape of the Weibull distribution. Default is 1, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only piecewiselambda (as a single value) and kappa can be specified. This function is equivalent to pweibull(t, shape = kappa, scale = 1 / lambda) of the stats package, i.e., the scale parameter is 1 / 'hazard rate'. For example, getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2) and pweibull(q = 130, shape = 4.2, scale = 1 / 0.01) provide the same result.
piecewiseSurvivalTime	A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function (for details see getPiecewiseSurvivalTime()).
lambda2	The assumed hazard rate in the reference group, there is no default. lambda2 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
lambda1	The assumed hazard rate in the treatment group, there is no default. lambda1 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. If allocationRatioPlanned = 0 is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.
hazardRatio	The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.
dropoutRate1	The assumed drop-out rate in the treatment group, default is 0.
dropoutRate2	The assumed drop-out rate in the control group, default is 0.
dropoutTime	The assumed time for drop-out rates in the control and the treatment group, default is 12.
maxNumberOfSubjects	If maxNumberOfSubjects > 0 is specified, the end of accrual at specified accrualIntensity for the specified number of subjects is determined or accrualIntensity is calculated at fixed end of accrual.

Details

The function computes the overall event probabilities in a two treatment groups design. For details of the parameters see `getSampleSizeSurvival()`.

Value

Returns a `EventProbabilities` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

Examples

```
## Not run:
# Calculate event probabilities for staggered subjects' entry, piecewisely defined
# survival time and hazards, and plot it.
timeVector <- seq(0, 100, 1)
y <- getEventProbabilities(timeVector, accrualTime = c(0, 20, 60),
    accrualIntensity = c(5, 20),
    piecewiseSurvivalTime = c(0, 20, 80),
    lambda2 = c(0.02, 0.06, 0.1),
    hazardRatio = 2
)
plot(timeVector, y$cumulativeEventProbabilities, type = 'l')

## End(Not run)
```

---

getFinalConfidenceInterval

Get Final Confidence Interval

Description

Returns the final confidence interval for the parameter of interest. It is based on the prototype case, i.e., the test for testing a mean for normally distributed variables.

Usage

```
getFinalConfidenceInterval(
  design,
  dataInput,
  ...,
  directionUpper = NA,
  thetaH0 = NA_real_,
  tolerance = 1e-06,
  stage = NA_integer_
)
```

Arguments

design	The trial design.
dataInput	The summary data used for calculating the test results. This is either an element of DatasetMeans, of DatasetRates, or of DatasetSurvival and should be created with the function getDataset(). For more information see getDataset().
...	Further (optional) arguments to be passed: normalApproximation The type of computation of the p-values. Default is FALSE for testing means (i.e., the t test is used) and TRUE for testing rates and the hazard ratio. For testing rates, if normalApproximation = FALSE is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting, normalApproximation = FALSE has no effect. equalVariances The type of t test. For testing means in two treatment groups, either the t test assuming that the variances are equal or the t test without assuming this, i.e., the test of Welch-Satterthwaite is calculated, default is TRUE. stdErrorEstimate Estimate of standard error for calculation of final confidence intervals for comparing rates in two treatment groups, default is "pooled".
directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.
thetaH0	The null hypothesis value, default is 0 for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio). For non-inferiority designs, thetaH0 is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value != 0 (or a value != 1 for testing the mean ratio) can be specified. • *rates*: a value != 0 (or a value != 1 for testing the risk ratio pi1 / pi2) can be specified. • *survival data*: a bound for testing H0: hazard ratio = thetaH0 != 1 can be specified. • *count data*: a bound for testing H0: lambda1 / lambda2 = thetaH0 != 1 can be specified. For testing a rate in one sample, a value thetaH0 in (0, 1) has to be specified for defining the null hypothesis H0: pi = thetaH0.

tolerance	The numerical tolerance, default is 1e-06. Must be a positive numeric of length 1.
stage	The stage number (optional). Default: total number of existing stages in the data input.

Details

Depending on design and dataInput the final confidence interval and median unbiased estimate that is based on the stage-wise ordering of the sample space will be calculated and returned. Additionally, a non-standardized ("general") version is provided, the estimated standard deviation must be used to obtain the confidence interval for the parameter of interest.

For the inverse normal combination test design with more than two stages, a warning informs that the validity of the confidence interval is theoretically shown only if no sample size change was performed.

Value

Returns a list containing

- finalStage,
- medianUnbiased,
- finalConfidenceInterval,
- medianUnbiasedGeneral, and
- finalConfidenceIntervalGeneral.

See Also

Other analysis functions: getAnalysisResults(), getClosedCombinationTestResults(), getClosedConditionalD
getConditionalPower(), getConditionalRejectionProbabilities(), getFinalPValue(), getRepeatedConfide
getRepeatedPValues(), getStageResults(), getTestActions()

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
data <- getDataset(
    n = c(20, 30),
    means = c(50, 51),
    stDevs = c(130, 140)
)
getFinalConfidenceInterval(design, dataInput = data)

## End(Not run)
```

`getFinalPValue` *Get Final P Value*

Description

Returns the final p-value for given stage results.

Usage

```
getFinalPValue(stageResults, ...)
```

Arguments

`stageResults`	The results at given stage, obtained from `getStageResults()`.
`...`	Only available for backward compatibility.

Details

The calculation of the final p-value is based on the stage-wise ordering of the sample space. This enables the calculation for both the non-adaptive and the adaptive case. For Fisher's combination test, it is available for `kMax = 2` only.

Value

Returns a `list` containing

- `finalStage`,
- `pFinal`.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedCombinationTestResults()`, `getClosedConditionalD`
`getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`,
`getRepeatedConfidenceIntervals()`, `getRepeatedPValues()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
data <- getDataset(
    n      = c( 20,  30),
    means  = c( 50,  51),
    stDevs = c(130, 140)
)
getFinalPValue(getStageResults(design, dataInput = data))

## End(Not run)
```

getFutilityBounds *Get Futility Bounds*

Description

This function converts futility bounds between different scales such as z-value, p-value, conditional power, predictive power, reverse conditional power, and effect estimate.

Usage

```
getFutilityBounds(
  sourceValue,
  ...,
  sourceScale = c("zValue", "pValue", "conditionalPower", "condPowerAtObserved",
    "predictivePower", "reverseCondPower", "effectEstimate"),
  targetScale = c("zValue", "pValue", "conditionalPower", "condPowerAtObserved",
    "predictivePower", "reverseCondPower", "effectEstimate"),
  design = NULL,
  theta = NA_real_,
  information = NA_real_,
  naAllowed = FALSE
)
```

Arguments

sourceValue	A numeric vector representing the futility bounds in the source scale.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
sourceScale	Character. The scale of the input futility bounds. Must be one of "zValue", "pValue", "conditionalPower", "condPowerAtObserved", "predictivePower", "reverseCondPower", or "effectEstimate".
targetScale	Character. The scale to which the futility bounds should be converted. Must be one of "zValue", "pValue", "conditionalPower", "condPowerAtObserved", "predictivePower", "reverseCondPower", or "effectEstimate".
design	The trial design. Required if either the sourceScale or targetScale is "reverseCondPower" or if the conversion involves conditional or predictive power in a group sequential or Fisher design. Must be a one-sided two-stage group sequential design or Fisher's combination test design.
theta	Numeric. The assumed effect size under the alternative hypothesis.
information	Numeric vector of length 2. The information levels at the two stages.
naAllowed	Logical. Indicates if NA sourceValue are permitted. Default is FALSE.

Details

If the sourceScale and targetScale are the same, the function returns the input sourceValue without modification. Otherwise, the function is designed to convert between the specified scales.

Value

A numeric vector representing the futility bounds in the target scale, or NULL if the conversion is not implemented or yields no result.

See Also

getDesignGroupSequential(), getDesignInverseNormal(), getDesignFisher() for direct specification of futility bounds on different scales using the argument futilityBoundsScale.

Examples

```
## Not run:
# Example with identical source and target scales
getFutilityBounds(
    sourceValue = c(0, 0.5),
    sourceScale = "zValue",
    targetScale = "zValue"
)

# Example with different scales
getFutilityBounds(
   design = getDesignGroupSequential(kMax = 2, typeOfDesign = "noEarlyEfficacy", alpha = 0.05),
    information = c(10, 10),
    sourceValue = 0.5,
    sourceScale = "condPowerAtObserved",
    targetScale = "pValue"
)

## End(Not run)
```

---

getGroupSequentialProbabilities

Get Group Sequential Probabilities

Description

Calculates probabilities in the group sequential setting.

Usage

```
getGroupSequentialProbabilities(decisionMatrix, informationRates)
```

Arguments

decisionMatrix	A matrix with either 2 or 4 rows and kMax = length(informationRates) columns, see details.
informationRates	The information rates t_1, ..., t_kMax (that must be fixed prior to the trial), default is (1:kMax) / kMax. For the weighted inverse normal design, the weights are derived through w_1 = sqrt(t_1), and w_k = sqrt(t_k - t_(k-1)). For the weighted Fisher's combination test, the weights (scales) are w_k = sqrt((t_k - t_(k-1)) / t_1) (see the documentation).

Details

Given a sequence of information rates (fixing the correlation structure), and decisionMatrix with either 2 or 4 rows and kMax = length(informationRates) columns, this function calculates a probability matrix containing, for two rows, the probabilities:
P(Z_1 < l_1), P(l_1 < Z_1 < u_1, Z_2 < l_2),..., P(l_kMax-1 < Z_kMax-1 < u_kMax-1, Z_kMax < l_l_kMax)
P(Z_1 < u_1), P(l_1 < Z_1 < u_1, Z_2 < u_2),..., P(l_kMax-1 < Z_kMax-1 < u_kMax-1, Z_kMax < u_l_kMax)
P(Z_1 < Inf), P(l_1 < Z_1 < u_1, Z_2 < Inf),..., P(l_kMax-1 < Z_kMax-1 < u_kMax-1, Z_kMax < Inf)
with continuation matrix
l_1,...,l_kMax
u_1,...,u_kMax
That is, the output matrix of the function provides per stage (column) the cumulative probabilities for values specified in decisionMatrix and Inf, and reaching the stage, i.e., the test statistics is in the continuation region for the preceding stages. For 4 rows, the continuation region contains of two regions and the probability matrix is obtained analogously (cf., Wassmer and Brannath, 2025).

Value

Returns a numeric matrix containing the probabilities described in the details section.

See Also

Other design functions: `getDesignCharacteristics()`, `getDesignConditionalDunnett()`, `getDesignFisher()`, `getDesignGroupSequential()`, `getDesignInverseNormal()`, `getPowerAndAverageSampleNumber()`

Examples

```
## Not run:
# Calculate Type I error rates in the two-sided group sequential setting when
# performing kMax stages with constant critical boundaries at level alpha:
alpha <- 0.05
kMax <- 10
decisionMatrix <- matrix(c(
    rep(-qnorm(1 - alpha / 2), kMax),
    rep(qnorm(1 - alpha / 2), kMax)
), nrow = 2, byrow = TRUE)
informationRates <- (1:kMax) / kMax
probs <- getGroupSequentialProbabilities(decisionMatrix, informationRates)
cumsum(probs[3, ] - probs[2, ] + probs[1, ])

# Do the same for a one-sided design without futility boundaries:
decisionMatrix <- matrix(c(
    rep(-Inf, kMax),
    rep(qnorm(1 - alpha), kMax)
), nrow = 2, byrow = TRUE)
informationRates <- (1:kMax) / kMax
probs <- getGroupSequentialProbabilities(decisionMatrix, informationRates)
cumsum(probs[3, ] - probs[2, ])

# Check that two-sided Pampallona and Tsiatis boundaries with binding
# futility bounds obtain Type I error probabilities equal to alpha:
x <- getDesignGroupSequential(
    alpha = 0.05, beta = 0.1, kMax = 3, typeOfDesign = "PT",
```

```
    deltaPT0 = 0, deltaPT1 = 0.4, sided = 2, bindingFutility = TRUE
)
dm <- matrix(c(
    -x$criticalValues, -x$futilityBounds, 0,
    x$futilityBounds, 0, x$criticalValues
), nrow = 4, byrow = TRUE)
dm[is.na(dm)] <- 0
probs <- getGroupSequentialProbabilities(
    decisionMatrix = dm, informationRates = (1:3) / 3
)
sum(probs[5, ] - probs[4, ] + probs[1, ])

# Check the Type I error rate decrease when using non-binding futility bounds:
x <- getDesignGroupSequential(
    alpha = 0.05, beta = 0.1, kMax = 3, typeOfDesign = "PT",
    deltaPT0 = 0, deltaPT1 = 0.4, sided = 2, bindingFutility = FALSE
)
dm <- matrix(c(
    -x$criticalValues, -x$futilityBounds, 0,
    x$futilityBounds, 0, x$criticalValues
), nrow = 4, byrow = TRUE)
dm[is.na(dm)] <- 0
probs <- getGroupSequentialProbabilities(
    decisionMatrix = dm, informationRates = (1:3) / 3
)
sum(probs[5, ] - probs[4, ] + probs[1, ])

## End(Not run)
```

`getLambdaStepFunction` *Get Lambda Step Function*

Description

Calculates the lambda step values for a given time vector.

Usage

```
getLambdaStepFunction(timeValues, ..., piecewiseSurvivalTime, piecewiseLambda)
```

Arguments

`timeValues`	A numeric vector that specifies the time values for which the lambda step values shall be calculated.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`piecewiseSurvivalTime`	A numeric vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function (see details).
`piecewiseLambda`	A numeric vector that specifies the assumed hazard rate in the treatment group.

Details

The first element of the vector `piecewiseSurvivalTime` must be equal to `0`. This function is used for plotting of sample size survival results (cf., `plot`, `type = 13` and `type = 14`).

Value

A numeric vector containing the lambda step values that corresponds to the specified time values.

`getLogLevel` *Get Log Level*

Description

Returns the current `rpact` log level.

Usage

```
getLogLevel()
```

Details

This function gets the log level of the `rpact` internal log message system.

Value

Returns a `character` of length 1 specifying the current log level.

See Also

- `setLogLevel()` for setting the log level,
- `resetLogLevel()` for resetting the log level to default.

Examples

```
# show current log level
getLogLevel()
```

getLongFormat *Get Long Format*

Description

Returns the specified dataset as a `data.frame` in so-called long format.

Usage

```
getLongFormat(dataInput)
```

Details

In the long format (narrow, stacked), the data are presented with one column containing all the values and another column listing the context of the value, i.e., the data for the different groups are in one column and the dataset contains an additional "group" column.

Value

A `data.frame` will be returned.

See Also

`getWideFormat()` for returning the dataset as a `data.frame` in wide format.

getNumberOfSubjects *Get Number Of Subjects*

Description

Returns the number of recruited subjects at given time vector.

Usage

```
getNumberOfSubjects(
  time,
  ...,
  accrualTime = c(0, 12),
  accrualIntensity = 0.1,
  accrualIntensityType = c("auto", "absolute", "relative"),
  maxNumberOfSubjects = NA_real_
)
```

Arguments

`time`	A numeric vector with time values.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`accrualTime`	The assumed accrual time intervals for the study, default is `c(0, 12)` (for details see `getAccrualTime()`).
`accrualIntensity`	A numeric vector of accrual intensities, default is the relative intensity `0.1` (for details see `getAccrualTime()`).
`accrualIntensityType`	A character value specifying the accrual intensity input type. Must be one of `"auto"`, `"absolute"`, or `"relative"`; default is `"auto"`, i.e., if all values are < 1 the type is `"relative"`, otherwise it is `"absolute"`.
`maxNumberOfSubjects`	If `maxNumberOfSubjects > 0` is specified, the end of accrual at specified `accrualIntensity` for the specified number of subjects is determined or `accrualIntensity` is calculated at fixed end of accrual.

Details

Calculate number of subjects over time range at given accrual time vector and accrual intensity. Intensity can either be defined in absolute or relative terms (for the latter, `maxNumberOfSubjects` needs to be defined)
The function is used by `getSampleSizeSurvival()`.

Value

Returns a `NumberOfSubjects` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

See Also

`AccrualTime` for defining the accrual time.

Examples

```
## Not run:
getNumberOfSubjects(time = seq(10, 70, 10), accrualTime = c(0, 20, 60),
    accrualIntensity = c(5, 20))

getNumberOfSubjects(time = seq(10, 70, 10), accrualTime = c(0, 20, 60),
    accrualIntensity = c(0.1, 0.4), maxNumberOfSubjects = 900)

## End(Not run)
```

---

getObservedInformationRates

Get Observed Information Rates

Description

Recalculates the observed information rates from the specified dataset.

Usage

```
getObservedInformationRates(
  dataInput,
  ...,
  maxInformation = NULL,
  informationEpsilon = NULL,
  stage = NA_integer_
)
```

Arguments

`dataInput`	The dataset for which the information rates shall be recalculated.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`maxInformation`	Positive value specifying the maximum information.
`informationEpsilon`	Positive integer value specifying the absolute information epsilon, which defines the maximum distance from the observed information to the maximum information that causes the final analysis. Updates at the final analysis in case the observed information at the final analysis is smaller ("under-running") than the planned maximum information `maxInformation`, default is 0. Alternatively, a floating-point number > 0 and < 1 can be specified to define a relative information epsilon.
`stage`	The stage number (optional). Default: total number of existing stages in the data input.

Details

For means and rates the maximum information is the maximum number of subjects or the relative proportion if `informationEpsilon` < 1; for survival data it is the maximum number of events or the relative proportion if `informationEpsilon` < 1.

Value

Returns a list that summarizes the observed information rates.

See Also

- `getAnalysisResults()` for using `getObservedInformationRates()` implicit,
- www.rpact.org/vignettes/planning/rpact_boundary_update_example

Examples

```
## Not run:
# Absolute information epsilon:
# decision rule 45 >= 46 - 1, i.e., under-running
data <- getDataset(
    overallN = c(22, 45),
    overallEvents = c(11, 28)
)
getObservedInformationRates(data,
    maxInformation = 46, informationEpsilon = 1
)

# Relative information epsilon:
# last information rate = 45/46 = 0.9783,
# is > 1 - 0.03 = 0.97, i.e., under-running
data <- getDataset(
    overallN = c(22, 45),
    overallEvents = c(11, 28)
)
getObservedInformationRates(data,
    maxInformation = 46, informationEpsilon = 0.03
)

## End(Not run)
```

---

getOutputFormat *Get Output Format*

Description

With this function the format of the standard outputs of all `rpact` objects can be shown and written to a file.

Usage

```
getOutputFormat(
  parameterName = NA_character_,
  ...,
  file = NA_character_,
  default = FALSE,
  fields = TRUE
)
```

Arguments

parameterName	The name of the parameter whose output format shall be returned. Leave the default NA_character_ if the output format of all parameters shall be returned.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
file	An optional file name where to write the output formats (see Details for more information).
default	If TRUE the default output format of the specified parameter(s) will be returned, default is FALSE.
fields	If TRUE the names of all affected object fields will be displayed, default is TRUE.

Details

Output formats can be written to a text file by specifying a file. See setOutputFormat()() to learn how to read a formerly saved file.

Note that the parameterName must not match exactly, e.g., for p-values the following parameter names will be recognized amongst others:

1. p value
2. p.values
3. p-value
4. pValue
5. rpact.output.format.p.value

Value

A named list of output formats.

See Also

Other output formats: setOutputFormat()

Examples

```
## Not run:
# show output format of p values
getOutputFormat("p.value")

# set new p value output format
setOutputFormat("p.value", digits = 5, nsmall = 5)

# show sample sizes as smallest integers not less than the not rounded values
setOutputFormat("sample size", digits = 0, nsmall = 0, roundFunction = "ceiling")
getSampleSizeMeans()

# show sample sizes as smallest integers not greater than the not rounded values
setOutputFormat("sample size", digits = 0, nsmall = 0, roundFunction = "floor")
getSampleSizeMeans()

# set new sample size output format without round function
setOutputFormat("sample size", digits = 2, nsmall = 2)
getSampleSizeMeans()
```

```
# reset sample size output format to default
setOutputFormat("sample size")
getSampleSizeMeans()
getOutputFormat("sample size")

## End(Not run)
```

`getParameterCaption` *Get Parameter Caption*

Description

Returns the parameter caption for a given object and parameter name.

Usage

```
getParameterCaption(obj, var)
```

Arguments

`obj`	The rpact result object.
`var`	The variable/parameter name.

Details

This function identifies and returns the caption that will be used in print outputs of an rpact result object.

Value

Returns a `character` of specifying the corresponding caption of a given parameter name. Returns `NULL` if the specified `parameterName` does not exist.

See Also

`getParameterName()` for getting the parameter name for a given caption.

Examples

```
## Not run:
getParameterCaption(getDesignInverseNormal(), "kMax")

## End(Not run)
```

getParameterName *Get Parameter Name*

Description

Returns the parameter name for a given object and parameter caption.

Usage

```
getParameterName(obj, parameterCaption)
```

Arguments

obj	The rpact result object.
parameterCaption	The parameter caption.

Details

This function identifies and returns the parameter name for a given caption that will be used in print outputs of an rpact result object.

Value

Returns a character of specifying the corresponding name of a given parameter caption. Returns NULL if the specified parameterCaption does not exist.

See Also

getParameterCaption() for getting the parameter caption for a given name.

Examples

```
## Not run:
getParameterName(getDesignInverseNormal(), "Maximum number of stages")

## End(Not run)
```

getParameterType *Get Parameter Type*

Description

Returns the parameter type for a given object and parameter name.

Usage

```
getParameterType(obj, var)
```

Arguments

obj	The rpact result object.
var	The variable/parameter name.

Details

This function identifies and returns the type that will be used in print outputs of an rpact result object.

Value

Returns a character of specifying the corresponding type of a given parameter name. Returns NULL if the specified parameterName does not exist.

See Also

getParameterName() for getting the parameter name for a given caption. getParameterCaption() for getting the parameter caption for a given name.

Examples

```
## Not run:
getParameterType(getDesignInverseNormal(), "kMax")

## End(Not run)
```

---

getPerformanceScore *Get Performance Score*

Description

Calculates the conditional performance score, its sub-scores and components according to (Herrmann et al. (2020), doi:10.1002/sim.8534) and (Bokelmann et al. (2024), doi:10.1186/s12874024-021504) for a given simulation result from a two-stage design with continuous or binary endpoint. Larger (sub-)score and component values refer to a better performance.

Usage

```
getPerformanceScore(simulationResult)
```

Arguments

simulationResult	A simulation result.

Details

The conditional performance score consists of two sub-scores, one for the sample size (subscoreSampleSize) and one for the conditional power (subscoreConditionalPower). Each of those are composed of a location (locationSampleSize, locationConditionalPower) and variation component (variationSampleSize, variationConditionalPower). The term conditional refers to an evaluation perspective where the interim results suggest a trial continuation with a second stage. The score can take values between 0 and 1. More details on the performance score can be found in Herrmann et al. (2020), doi:10.1002/sim.8534 and Bokelmann et al. (2024) doi:10.1186/s12874024021504.

Author(s)

Stephen Schueuerhuis

Examples

```
## Not run:
# Example from Table 3 in "A new conditional performance score for
# the evaluation of adaptive group sequential designs with samplesize
# recalculation from Herrmann et al 2023", p. 2097 for
# Observed Conditional Power approach and Delta = 0.5

# Create two-stage Pocock design with binding futility boundary at 0
design <- getDesignGroupSequential(
    kMax = 2, typeOfDesign = "P",
    futilityBounds = 0, bindingFutility = TRUE)

# Initialize sample sizes and effect;
# Sample sizes are referring to overall stage-wise sample sizes
n1 <- 100
n2 <- 100
nMax <- n1 + n2
alternative <- 0.5

# Perform Simulation; nMax * 1.5 defines the maximum
# sample size for the additional stage
simulationResult <- getSimulationMeans(
    design = design,
    normalApproximation = TRUE,
    thetaH0 = 0,
    alternative = alternative,
    plannedSubjects = c(n1, nMax),
    minNumberOfSubjectsPerStage = c(NA_real_, 1),
    maxNumberOfSubjectsPerStage = c(NA_real_, nMax * 1.5),
    conditionalPower = 0.8,
    directionUpper = TRUE,
    maxNumberOfIterations = 1e05,
    seed = 140
)

# Calculate performance score
getPerformanceScore(simulationResult)

## End(Not run)
```

getPiecewiseSurvivalTime
Get Piecewise Survival Time

Description

Returns a PiecewiseSurvivalTime object that contains the all relevant parameters of an exponential survival time cumulative distribution function. Use names to obtain the field names.

Usage

```
getPiecewiseSurvivalTime(
  piecewiseSurvivalTime = NA_real_,
  ...,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  hazardRatio = NA_real_,
  pi1 = NA_real_,
  pi2 = NA_real_,
  median1 = NA_real_,
  median2 = NA_real_,
  eventTime = 12,
  kappa = 1,
  delayedResponseAllowed = FALSE
)
```

Arguments

piecewiseSurvivalTime	A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function (see details).
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
lambda1	The assumed hazard rate in the treatment group, there is no default. lambda1 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
lambda2	The assumed hazard rate in the reference group, there is no default. lambda2 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
hazardRatio	The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.
pi1	A numeric value or vector that represents the assumed event rate in the treatment group, default is seq(0.2, 0.5, 0.1) (power calculations and simulations) or seq(0.4, 0.6, 0.1) (sample size calculations).
pi2	A numeric value that represents the assumed event rate in the control group, default is 0.2.
median1	The assumed median survival time in the treatment group, there is no default.

median2 The assumed median survival time in the reference group, there is no default. Must be a positive numeric of length 1.

eventTime The assumed time under which the event rates are calculated, default is 12.

kappa A numeric value > 0. A kappa != 1 will be used for the specification of the shape of the Weibull distribution. Default is 1, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only piecewiselambda (as a single value) and kappa can be specified. This function is equivalent to pweibull(t, shape = kappa, scale = 1 / lambda) of the stats package, i.e., the scale parameter is 1 / 'hazard rate'.
For example, getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2) and pweibull(q = 130, shape = 4.2, scale = 1 / 0.01) provide the same result.

delayedResponseAllowed
If TRUE, delayed response is allowed; otherwise it will be validated that the response is not delayed, default is FALSE.

Value

Returns a PiecewiseSurvivalTime object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Piecewise survival time

The first element of the vector piecewiseSurvivalTime must be equal to 0. piecewiseSurvivalTime can also be a list that combines the definition of the time intervals and hazard rates in the reference group. The definition of the survival time in the treatment group is obtained by the specification of the hazard ratio (see examples for details).

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
getPiecewiseSurvivalTime(lambda2 = 0.5, hazardRatio = 0.8)

getPiecewiseSurvivalTime(lambda2 = 0.5, lambda1 = 0.4)

getPiecewiseSurvivalTime(pi2 = 0.5, hazardRatio = 0.8)
```

```
getPiecewiseSurvivalTime(pi2 = 0.5, pi1 = 0.4)

getPiecewiseSurvivalTime(pi1 = 0.3)

getPiecewiseSurvivalTime(hazardRatio = c(0.6, 0.8), lambda2 = 0.4)

getPiecewiseSurvivalTime(piecewiseSurvivalTime = c(0, 6, 9),
    lambda2 = c(0.025, 0.04, 0.015), hazardRatio = 0.8)

getPiecewiseSurvivalTime(piecewiseSurvivalTime = c(0, 6, 9),
    lambda2 = c(0.025, 0.04, 0.015),
    lambda1 = c(0.025, 0.04, 0.015) * 0.8)

pwst <- getPiecewiseSurvivalTime(list(
    "0 - <6"   = 0.025,
    "6 - <9"   = 0.04,
    "9 - <15"  = 0.015,
    "15 - <21" = 0.01,
    ">=21"     = 0.007), hazardRatio = 0.75)
pwst

# The object created by getPiecewiseSurvivalTime() can be used directly in
# getSampleSizeSurvival():
getSampleSizeSurvival(piecewiseSurvivalTime = pwst)

# The object created by getPiecewiseSurvivalTime() can be used directly in
# getPowerSurvival():
getPowerSurvival(piecewiseSurvivalTime = pwst, directionUpper = FALSE,
    maxNumberOfEvents = 40, maxNumberOfSubjects = 100)

# The object created by getPiecewiseSurvivalTime() can be used directly in
# getSimulationSurvival():
getSimulationSurvival(piecewiseSurvivalTime = pwst, directionUpper = FALSE,
    plannedEvents = 40, maxNumberOfSubjects = 100)

## End(Not run)
```

`getPlotSettings` *Get Plot Settings*

Description

Returns a plot settings object.

Usage

```
getPlotSettings(
  lineSize = 0.8,
  pointSize = 3,
  pointColor = NA_character_,
  mainTitleFontSize = 14,
  axesTextFontSize = 10,
```

```
  legendFontSize = 11,
  scalingFactor = 1
)
```

Arguments

lineSize	The line size, default is 0.8.
pointSize	The point size, default is 3.
pointColor	The point color (character), default is NA_character_.
mainTitleFontSize	The main title font size, default is 14.
axesTextFontSize	The axes text font size, default is 10.
legendFontSize	The legend font size, default is 11.
scalingFactor	The scaling factor, default is 1.

Details

Returns an object of class PlotSettings that collects typical plot settings.

---

getPowerAndAverageSampleNumber

Get Power And Average Sample Number

Description

Returns the power and average sample number of the specified design.

Usage

```
getPowerAndAverageSampleNumber(design, theta = seq(-1, 1, 0.02), nMax = 100)
```

Arguments

design	The trial design.
theta	A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.
nMax	

Value

Returns a PowerAndAverageSampleNumberResult object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other design functions: getDesignCharacteristics(), getDesignConditionalDunnett(), getDesignFisher(), getDesignGroupSequential(), getDesignInverseNormal(), getGroupSequentialProbabilities()

Examples

```
## Not run:
# Calculate power, stopping probabilities, and expected sample
# size for the default design with specified theta and nMax
getPowerAndAverageSampleNumber(
    getDesignGroupSequential(),
    theta = seq(-1, 1, 0.5), nMax = 100)

## End(Not run)
```

---

getPowerCounts *Get Power Counts*

Description

Returns the power, stopping probabilities, and expected sample size for testing mean rates for negative binomial distributed event numbers in two samples at given sample sizes.

Usage

```
getPowerCounts(
  design = NULL,
  ...,
  directionUpper = NA,
  maxNumberOfSubjects = NA_real_,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  lambda = NA_real_,
  theta = NA_real_,
  thetaH0 = 1,
  overdispersion = 0,
  fixedExposureTime = NA_real_,
  accrualTime = NA_real_,
  accrualIntensity = NA_real_,
  followUpTime = NA_real_,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`directionUpper`	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.
`maxNumberOfSubjects`	`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.
`lambda1`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the active treatment group, there is no default.
`lambda2`	A numeric value that represents the assumed rate of a homogeneous Poisson process in the control group, there is no default.
`lambda`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the pooled treatment groups, there is no default.
`theta`	A numeric value or vector that represents the assumed mean ratios lambda1/lambda2 of a homogeneous Poisson process, there is no default.
`thetaH0`	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.

- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`overdispersion` A numeric value that represents the assumed overdispersion of the negative binomial distribution, default is `0`.

`fixedExposureTime`
If specified, the fixed time of exposure per subject for count data, there is no default.

`accrualTime` If specified, the assumed accrual time interval(s) for the study, there is no default.

`accrualIntensity`
If specified, the assumed accrual intensities for the study, there is no default.

`followUpTime` If specified, the assumed (additional) follow-up time for the study, there is no default. The total study duration is `accrualTime + followUpTime`.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

Details

At given design the function calculates the power, stopping probabilities, and expected sample size for testing the ratio of two mean rates of negative binomial distributed event numbers in two samples at given maximum sample size and effect. The power calculation is performed either for a fixed exposure time or a variable exposure time with fixed follow-up where the information over the stages is calculated according to the specified information rate in the design. Additionally, an allocation ratio = `n1 / n2` can be specified where `n1` and `n2` are the number of subjects in the two treatment groups. A null hypothesis value `thetaH0` can also be specified.

Value

Returns a `TrialDesignPlan` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other power functions: getPowerMeans(), getPowerRates(), getPowerSurvival()

Examples

```
## Not run:
# Fixed sample size trial where a therapy is assumed to decrease the
# exacerbation rate from 1.4 to 1.05 (25% decrease) within an
# observation period of 1 year, i.e., each subject has a equal
# follow-up of 1 year.
# Calculate power at significance level 0.025 at given sample size = 180
# for a range of lambda1 values if the overdispersion is assumed to be
# equal to 0.5, is obtained by
getPowerCounts(alpha = 0.025, lambda1 = seq(1, 1.4, 0.05), lambda2 = 1.4,
    maxNumberOfSubjects = 180, overdispersion = 0.5, fixedExposureTime = 1)

# Group sequential alpha and beta spending function design with O'Brien and
# Fleming type boundaries: Power and test characteristics for N = 286,
# under the assumption of a fixed exposure time, and for a range of
# lambda1 values:
getPowerCounts(design = getDesignGroupSequential(
        kMax = 3, alpha = 0.025, beta = 0.2,
        typeOfDesign = "asOF", typeBetaSpending = "bsOF"),
    lambda1 = seq(0.17, 0.23, 0.01), lambda2 = 0.3,
    directionUpper = FALSE, overdispersion = 1, maxNumberOfSubjects = 286,
    fixedExposureTime = 12, accrualTime = 6)

# Group sequential design alpha spending function design with O'Brien and
# Fleming type boundaries: Power and test characteristics for N = 1976,
# under variable exposure time with uniform recruitment over 1.25 months,
# study time (accrual + followup) = 4 (lambda1, lambda2, and overdispersion
# as specified, no futility stopping):
getPowerCounts(design = getDesignGroupSequential(
        kMax = 3, alpha = 0.025, beta = 0.2, typeOfDesign = "asOF"),
    lambda1 = seq(0.08, 0.09, 0.0025), lambda2 = 0.125,
    overdispersion = 5, directionUpper = FALSE, maxNumberOfSubjects = 1976,
    followUpTime = 2.75, accrualTime = 1.25)

## End(Not run)
```

---

getPowerMeans *Get Power Means*

Description

Returns the power, stopping probabilities, and expected sample size for testing means in one or two samples at given maximum sample size.

Usage

```
getPowerMeans(
  design = NULL,
  ...,
  groups = 2L,
  normalApproximation = FALSE,
  meanRatio = FALSE,
  thetaH0 = ifelse(meanRatio, 1, 0),
  alternative = seq(0, 1, 0.2),
  stDev = 1,
  directionUpper = NA,
  maxNumberOfSubjects = NA_real_,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`groups`	The number of treatment groups (1 or 2), default is `2`.
`normalApproximation`	The type of computation of the p-values. If `TRUE`, the variance is assumed to be known, default is `FALSE`, i.e., the calculations are performed with the t distribution.
`meanRatio`	If `TRUE`, the sample size for one-sided testing of H0: `mu1 / mu2 = thetaH0` is calculated, default is `FALSE`.
`thetaH0`	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified. • *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified. • *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified. • *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified. For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`alternative` The alternative hypothesis value for testing means. This can be a vector of assumed alternatives, default is `seq(0, 1, 0.2)` (power calculations) or `seq(0.2, 1, 0.2)` (sample size calculations).

`stDev` The standard deviation under which the sample size or power calculation is performed, default is `1`. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If `meanRatio = TRUE` is specified, `stDev` defines the coefficient of variation `sigma / mu2`.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`maxNumberOfSubjects`
`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

Details

At given design the function calculates the power, stopping probabilities, and expected sample size for testing means at given sample size. In a two treatment groups design, additionally, an allocation ratio = `n1 / n2` can be specified where `n1` and `n2` are the number of subjects in the two treatment groups. A null hypothesis value thetaH0 != 0 for testing the difference of two means or `thetaH0 != 1` for testing the ratio of two means can be specified. For the specified sample size, critical bounds and stopping for futility bounds are provided at the effect scale (mean, mean difference, or mean ratio, respectively)

Value

Returns a `TrialDesignPlan` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")`

to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other power functions: getPowerCounts(), getPowerRates(), getPowerSurvival()

Examples

```
## Not run:
# Calculate the power, stopping probabilities, and expected sample size
# for testing H0: mu1 - mu2 = 0 in a two-armed design against a range of
# alternatives H1: mu1 - m2 = delta, delta = (0, 1, 2, 3, 4, 5),
# standard deviation sigma = 8, maximum sample size N = 80 (both treatment
# arms), and an allocation ratio n1/n2 = 2. The design is a three stage
# O'Brien & Fleming design with non-binding futility bounds (-0.5, 0.5)
# for the two interims. The computation takes into account that the t test
# is used (normalApproximation = FALSE).
getPowerMeans(getDesignGroupSequential(alpha = 0.025,
    sided = 1, futilityBounds = c(-0.5, 0.5)),
    groups = 2, alternative = c(0:5), stDev = 8,
    normalApproximation = FALSE, maxNumberOfSubjects = 80,
    allocationRatioPlanned = 2)

## End(Not run)
```

---

getPowerRates *Get Power Rates*

Description

Returns the power, stopping probabilities, and expected sample size for testing rates in one or two samples at given maximum sample size.

Usage

```
getPowerRates(
  design = NULL,
  ...,
  groups = 2L,
  riskRatio = FALSE,
  thetaH0 = ifelse(riskRatio, 1, 0),
  pi1 = seq(0.2, 0.5, 0.1),
  pi2 = 0.2,
  directionUpper = NA,
  maxNumberOfSubjects = NA_real_,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design` The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.

`...` Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

`groups` The number of treatment groups (1 or 2), default is `2`.

`riskRatio` If `TRUE`, the power for one-sided testing of H0: `pi1 / pi2 = thetaH0` is calculated, default is `FALSE`.

`thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio).

For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`pi1` A numeric value or vector that represents the assumed probability in the active treatment group if two treatment groups are considered, or the alternative probability for a one treatment group design, default is `seq(0.2, 0.5, 0.1)` (power calculations and simulations) or `seq(0.4, 0.6, 0.1)` (sample size calculations).

`pi2` A numeric value that represents the assumed probability in the reference group if two treatment groups are considered, default is `0.2`.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`maxNumberOfSubjects`
`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

Details

At given design the function calculates the power, stopping probabilities, and expected sample size for testing rates at given maximum sample size. The sample sizes over the stages are calculated according to the specified information rate in the design. In a two treatment groups design, additionally, an allocation ratio = n1 / n2 can be specified where n1 and n2 are the number of subjects in the two treatment groups. If a null hypothesis value thetaH0 != 0 for testing the difference of two rates or thetaH0 != 1 for testing the risk ratio is specified, the formulas according to Farrington & Manning (Statistics in Medicine, 1990) are used (only one-sided testing). Critical bounds and stopping for futility bounds are provided at the effect scale (rate, rate difference, or rate ratio, respectively). For the two-sample case, the calculation here is performed at fixed pi2 as given as argument in the function. Note that the power calculation for rates is always based on the normal approximation.

Value

Returns a TrialDesignPlan object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other power functions: getPowerCounts(), getPowerMeans(), getPowerSurvival()

Examples

```
## Not run:
# Calculate the power, stopping probabilities, and expected sample size in a
# two-armed design at given maximum sample size N = 200 in a three-stage
# O'Brien & Fleming design with information rate vector (0.2,0.5,1),
# non-binding futility boundaries (0,0), i.e., the study stops for futility
# if the p-value exceeds 0.5 at interm, and allocation ratio = 2 for a range
# of pi1 values when testing H0: pi1 - pi2 = -0.1:
getPowerRates(getDesignGroupSequential(informationRates = c(0.2, 0.5, 1),
    futilityBounds = c(0, 0)), groups = 2, thetaH0 = -0.1,
    pi1 = seq(0.3, 0.6, 0.1), directionUpper = FALSE,
    pi2 = 0.7, allocationRatioPlanned = 2, maxNumberOfSubjects = 200)

# Calculate the power, stopping probabilities, and expected sample size in a single
# arm design at given maximum sample size N = 60 in a three-stage two-sided
```

```
# O'Brien & Fleming design with information rate vector (0.2, 0.5,1)
# for a range of pi1 values when testing H0: pi = 0.3:
getPowerRates(getDesignGroupSequential(informationRates = c(0.2, 0.5,1),
    sided = 2), groups = 1, thetaH0 = 0.3, pi1 = seq(0.3, 0.5, 0.05),
    maxNumberOfSubjects = 60)

## End(Not run)
```

`getPowerSurvival` *Get Power Survival*

Description

Returns the power, stopping probabilities, and expected sample size for testing the hazard ratio in a two treatment groups survival design.

Usage

```
getPowerSurvival(
  design = NULL,
  ...,
  typeOfComputation = c("Schoenfeld", "Freedman", "HsiehFreedman"),
  thetaH0 = 1,
  directionUpper = NA,
  pi1 = NA_real_,
  pi2 = NA_real_,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  median1 = NA_real_,
  median2 = NA_real_,
  kappa = 1,
  hazardRatio = NA_real_,
  piecewiseSurvivalTime = NA_real_,
  allocationRatioPlanned = 1,
  eventTime = 12,
  accrualTime = c(0, 12),
  accrualIntensity = 0.1,
  accrualIntensityType = c("auto", "absolute", "relative"),
  maxNumberOfSubjects = NA_real_,
  maxNumberOfEvents = NA_real_,
  dropoutRate1 = 0,
  dropoutRate2 = 0,
  dropoutTime = 12
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.

`...` Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

`typeOfComputation`
Three options are available: `"Schoenfeld"`, `"Freedman"`, `"HsiehFreedman"`, the default is `"Schoenfeld"`. For details, see Hsieh (Statistics in Medicine, 1992). For non-inferiority testing (i.e., `thetaH0 != 1`), only Schoenfeld's formula can be used.

`thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio).

For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`pi1` A numeric value or vector that represents the assumed event rate in the treatment group, default is `seq(0.2, 0.5, 0.1)` (power calculations and simulations) or `seq(0.4, 0.6, 0.1)` (sample size calculations).

`pi2` A numeric value that represents the assumed event rate in the control group, default is `0.2`.

`lambda1` The assumed hazard rate in the treatment group, there is no default. `lambda1` can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

`lambda2` The assumed hazard rate in the reference group, there is no default. `lambda2` can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

`median1` The assumed median survival time in the treatment group, there is no default.

`median2` The assumed median survival time in the reference group, there is no default. Must be a positive numeric of length 1.

`kappa` A numeric value > 0. A `kappa != 1` will be used for the specification of the shape of the Weibull distribution. Default is 1, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only `piecewiselambda` (as a single value) and `kappa` can be specified. This function is equivalent to `pweibull(t, shape = kappa, scale = 1 / lambda)` of the `stats` package, i.e., the scale parameter is `1 / 'hazard rate'`.
For example, `getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2)` and `pweibull(q = 130, shape = 4.2, scale = 1 / 0.01)` provide the same result.

hazardRatio The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.

piecewiseSurvivalTime
A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function
(for details see getPiecewiseSurvivalTime()).

allocationRatioPlanned
The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

eventTime The assumed time under which the event rates are calculated, default is 12.

accrualTime The assumed accrual time intervals for the study, default is c(0, 12) (for details see getAccrualTime()).

accrualIntensity
A numeric vector of accrual intensities, default is the relative intensity 0.1 (for details see getAccrualTime()).

accrualIntensityType
A character value specifying the accrual intensity input type. Must be one of "auto", "absolute", or "relative"; default is "auto", i.e., if all values are < 1 the type is "relative", otherwise it is "absolute".

maxNumberOfSubjects
maxNumberOfSubjects > 0 needs to be specified. If accrual time and accrual intensity are specified, this will be calculated. Must be a positive integer of length 1.

maxNumberOfEvents
maxNumberOfEvents > 0 is the maximum number of events, it determines the power of the test and needs to be specified.

dropoutRate1 The assumed drop-out rate in the treatment group, default is 0.

dropoutRate2 The assumed drop-out rate in the control group, default is 0.

dropoutTime The assumed time for drop-out rates in the control and the treatment group, default is 12.

Details

At given design the function calculates the power, stopping probabilities, and expected sample size at given number of events and number of subjects. It also calculates the time when the required events are expected under the given assumptions (exponentially, piecewise exponentially, or Weibull distributed survival times and constant or non-constant piecewise accrual). Additionally, an allocation ratio = n1 / n2 can be specified where n1 and n2 are the number of subjects in the two treatment groups.

The formula of Kim & Tsiatis (Biometrics, 1990) is used to calculate the expected number of events under the alternative (see also Lakatos & Lan, Statistics in Medicine, 1992). These formulas are generalized to piecewise survival times and non-constant piecewise accrual over time.

Value

Returns a TrialDesignPlan object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Piecewise survival time

The first element of the vector piecewiseSurvivalTime must be equal to 0. piecewiseSurvivalTime can also be a list that combines the definition of the time intervals and hazard rates in the reference group. The definition of the survival time in the treatment group is obtained by the specification of the hazard ratio (see examples for details).

Staggered patient entry

accrualTime is the time period of subjects' accrual in a study. It can be a value that defines the end of accrual or a vector. In this case, accrualTime can be used to define a non-constant accrual over time. For this, accrualTime is a vector that defines the accrual intervals. The first element of accrualTime must be equal to 0 and, additionally, accrualIntensity needs to be specified. accrualIntensity itself is a value or a vector (depending on the length of accrualTime) that defines the intensity how subjects enter the trial in the intervals defined through accrualTime.

accrualTime can also be a list that combines the definition of the accrual time and accrual intensity (see below and examples for details).

If the length of accrualTime and the length of accrualIntensity are the same (i.e., the end of accrual is undefined), maxNumberOfSubjects > 0 needs to be specified and the end of accrual is calculated. In that case, accrualIntensity is the number of subjects per time unit, i.e., the absolute accrual intensity.

If the length of accrualTime equals the length of accrualIntensity - 1 (i.e., the end of accrual is defined), maxNumberOfSubjects is calculated if the absolute accrual intensity is given. If all elements in accrualIntensity are smaller than 1, accrualIntensity defines the *relative* intensity how subjects enter the trial. For example, accrualIntensity = c(0.1, 0.2) specifies that in the second accrual interval the intensity is doubled as compared to the first accrual interval. The actual (absolute) accrual intensity is calculated for the calculated or given maxNumberOfSubjects. Note that the default is accrualIntensity = 0.1 meaning that the *absolute* accrual intensity will be calculated.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other power functions: `getPowerCounts()`, `getPowerMeans()`, `getPowerRates()`

Examples

```
## Not run:
# Fixed sample size with minimum required definitions, pi1 = c(0.2, 0.3, 0.4, 0.5) and
# pi2 = 0.2 at event time 12, accrual time 12 and follow-up time 6 as default
getPowerSurvival(maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# Four stage O'Brien & Fleming group sequential design with minimum required
# definitions, pi1 = c(0.2, 0.3, 0.4, 0.5) and pi2 = 0.2 at event time 12,
# accrual time 12 and follow-up time 6 as default
getPowerSurvival(design = getDesignGroupSequential(kMax = 4),
    maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# For fixed sample design, determine necessary accrual time if 200 subjects and
# 30 subjects per time unit can be recruited
getPowerSurvival(maxNumberOfEvents = 40, accrualTime = c(0),
    accrualIntensity = 30, maxNumberOfSubjects = 200)

# Determine necessary accrual time if 200 subjects and if the first 6 time units
# 20 subjects per time unit can be recruited, then 30 subjects per time unit
getPowerSurvival(maxNumberOfEvents = 40, accrualTime = c(0, 6),
    accrualIntensity = c(20, 30), maxNumberOfSubjects = 200)

# Determine maximum number of Subjects if the first 6 time units 20 subjects per
# time unit can be recruited, and after 10 time units 30 subjects per time unit
getPowerSurvival(maxNumberOfEvents = 40, accrualTime = c(0, 6, 10),
    accrualIntensity = c(20, 30))

# Specify accrual time as a list
at <- list(
    "0 - <6"  = 20,
    "6 - Inf" = 30)
getPowerSurvival(maxNumberOfEvents = 40, accrualTime = at, maxNumberOfSubjects = 200)

# Specify accrual time as a list, if maximum number of subjects need to be calculated
at <- list(
    "0 - <6"   = 20,
    "6 - <=10" = 30)
getPowerSurvival(maxNumberOfEvents = 40, accrualTime = at)

# Specify effect size for a two-stage group design with O'Brien & Fleming boundaries
# Effect size is based on event rates at specified event time, directionUpper = FALSE
# needs to be specified because it should be shown that hazard ratio < 1
getPowerSurvival(design = getDesignGroupSequential(kMax = 2), pi1 = 0.2, pi2 = 0.3,
    eventTime = 24, maxNumberOfEvents = 40, maxNumberOfSubjects = 200,
    directionUpper = FALSE)

# Effect size is based on event rate at specified event time for the reference group
# and hazard ratio, directionUpper = FALSE needs to be specified
# because it should be shown that hazard ratio < 1
getPowerSurvival(design = getDesignGroupSequential(kMax = 2), hazardRatio = 0.5,
    pi2 = 0.3, eventTime = 24, maxNumberOfEvents = 40, maxNumberOfSubjects = 200,
    directionUpper = FALSE)
```

```
# Effect size is based on hazard rate for the reference group and hazard ratio,
# directionUpper = FALSE needs to be specified because it should be shown that
# hazard ratio < 1
getPowerSurvival(design = getDesignGroupSequential(kMax = 2), hazardRatio = 0.5,
    lambda2 = 0.02, maxNumberOfEvents = 40, maxNumberOfSubjects = 200,
    directionUpper = FALSE)

# Specification of piecewise exponential survival time and hazard ratios
getPowerSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    hazardRatio = c(1.5, 1.8, 2),  maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# Specification of piecewise exponential survival time as list and hazard ratios
pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04)
getPowerSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = pws, hazardRatio = c(1.5, 1.8, 2),
    maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# Specification of piecewise exponential survival time for both treatment arms
getPowerSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    lambda1 = c(0.015,0.03,0.06),  maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# Specification of piecewise exponential survival time as a list
pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04)
getPowerSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = pws, hazardRatio = c(1.5, 1.8, 2),
    maxNumberOfEvents = 40, maxNumberOfSubjects = 200)

# Specify effect size based on median survival times
getPowerSurvival(median1 = 5, median2 = 3,
    maxNumberOfEvents = 40, maxNumberOfSubjects = 200, directionUpper = FALSE)

# Specify effect size based on median survival times of
# Weibull distribtion with kappa = 2
getPowerSurvival(median1 = 5, median2 = 3, kappa = 2,
    maxNumberOfEvents = 40, maxNumberOfSubjects = 200, directionUpper = FALSE)

## End(Not run)
```

`getRawData` *Get Simulation Raw Data for Survival*

Description

Returns the raw survival data which was generated for simulation.

Usage

```
getRawData(x, aggregate = FALSE)
```

Arguments

x	A SimulationResults object created by getSimulationSurvival().
aggregate	Logical. If TRUE the raw data will be aggregated similar to the result of getData(), default is FALSE.

Details

This function works only if getSimulationSurvival() was called with a maxNumberOfRawDatasetsPerStage > 0 (default is 0).

This function can be used to get the simulated raw data from a simulation results object obtained by getSimulationSurvival(). Note that getSimulationSurvival() must called before with maxNumberOfRawDatasetsPerStage > 0. The data frame contains the following columns:

1. iterationNumber: The number of the simulation iteration.
2. stopStage: The stage of stopping.
3. subjectId: The subject id (increasing number 1, 2, 3, ...)
4. accrualTime: The accrual time, i.e., the time when the subject entered the trial.
5. treatmentGroup: The treatment group number (1 or 2).
6. survivalTime: The survival time of the subject.
7. dropoutTime: The dropout time of the subject (may be NA).
8. lastObservationTime: The specific observation time.
9. timeUnderObservation: The time under observation is defined as follows:

```
if (event == TRUE) {
    timeUnderObservation <- survivalTime
} else if (dropoutEvent == TRUE) {
    timeUnderObservation <- dropoutTime
} else {
    timeUnderObservation <- lastObservationTime - accrualTime
}
```

10. event: TRUE if an event occurred; FALSE otherwise.
11. dropoutEvent: TRUE if an dropout event occurred; FALSE otherwise.

Value

Returns a data.frame.

Examples

```
## Not run:
results <- getSimulationSurvival(
    pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, eventTime = 12,
    accrualTime = 24, plannedEvents = 40, maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50, maxNumberOfRawDatasetsPerStage = 5
)
rawData <- getRawData(results)
```

```
head(rawData)
dim(rawData)

## End(Not run)
```

`getRepeatedConfidenceIntervals` *Get Repeated Confidence Intervals*

Description

Calculates and returns the lower and upper limit of the repeated confidence intervals of the trial.

Usage

```
getRepeatedConfidenceIntervals(
  design,
  dataInput,
  ...,
  directionUpper = NA,
  tolerance = 1e-06,
  stage = NA_integer_
)
```

Arguments

`design`	The trial design.
`dataInput`	The summary data used for calculating the test results. This is either an element of `DatasetMeans`, of `DatasetRates`, or of `DatasetSurvival` and should be created with the function `getDataset()`. For more information see `getDataset()`.
`...`	Further arguments to be passed to methods (cf., separate functions in "See Also" below), e.g.,
	`normalApproximation` The type of computation of the p-values. Default is `FALSE` for testing means (i.e., the t test is used) and `TRUE` for testing rates and the hazard ratio. For testing rates, if `normalApproximation = FALSE` is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting, `normalApproximation = FALSE` has no effect.
	`equalVariances` The type of t test. For testing means in two treatment groups, either the t test assuming that the variances are equal or the t test without assuming this, i.e., the test of Welch-Satterthwaite is calculated, default is `TRUE`.
	`intersectionTest` Defines the multiple test for the intersection hypotheses in the closed system of hypotheses when testing multiple hypotheses. Five options are available in multi-arm designs: `"Dunnett"`, `"Bonferroni"`, `"Simes"`, `"Sidak"`, and `"Hierarchical"`, default is `"Dunnett"`. Four options are available in population enrichment designs: `"SpiessensDebois"` (one subset only), `"Bonferroni"`, `"Simes"`, and `"Sidak"`, default is `"Simes"`.

`varianceOption` Defines the way to calculate the variance in multiple treatment arms (> 2) or population enrichment designs for testing means. For multiple arms, three options are available: `"overallPooled"`, `"pairwisePooled"`, and `"notPooled"`, default is `"overallPooled"`. For enrichment designs, the options are: `"pooled"`, `"pooledFromFull"` (one subset only), and `"notPooled"`, default is `"pooled"`.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. For testing means and rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is `TRUE`.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`tolerance` The numerical tolerance, default is `1e-06`. Must be a positive numeric of length 1.

`stage` The stage number (optional). Default: total number of existing stages in the data input.

Details

The repeated confidence interval at a given stage of the trial contains the parameter values that are not rejected using the specified sequential design. It can be calculated at each stage of the trial and can thus be used as a monitoring tool.

The repeated confidence intervals are provided up to the specified stage.

Value

Returns a `matrix` with 2 rows and `kMax` columns containing the lower RCI limits in the first row and the upper RCI limits in the second row, where each column represents a stage.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedCombinationTestResults()`, `getClosedConditionalD`
`getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`,
`getFinalPValue()`, `getRepeatedPValues()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
data <- getDataset(
    n      = c( 20,  30),
    means  = c( 50,  51),
    stDevs = c(130, 140)
)
getRepeatedConfidenceIntervals(design, dataInput = data)

## End(Not run)
```

`getRepeatedPValues` *Get Repeated P Values*

Description

Calculates the repeated p-values for a given test results.

Usage

```
getRepeatedPValues(stageResults, ..., tolerance = 1e-06)
```

Arguments

`stageResults`	The results at given stage, obtained from `getStageResults()`.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`tolerance`	The numerical tolerance, default is `1e-06`. Must be a positive numeric of length 1.

Details

The repeated p-value at a given stage of the trial is defined as the smallest significance level under which at given test design the test results obtain rejection of the null hypothesis. It can be calculated at each stage of the trial and can thus be used as a monitoring tool.

The repeated p-values are provided up to the specified stage.

In multi-arm trials, the repeated p-values are defined separately for each treatment comparison within the closed testing procedure.

Value

Returns a `numeric` vector of length `kMax` or in case of multi-arm stage results a `matrix` (each column represents a stage, each row a comparison) containing the repeated p values.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedCombinationTestResults()`, `getClosedConditionalD`
`getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`,
`getFinalPValue()`, `getRepeatedConfidenceIntervals()`, `getStageResults()`, `getTestActions()`

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
data <- getDataset(
    n      = c( 20,  30),
    means  = c( 50,  51),
    stDevs = c(130, 140)
)
getRepeatedPValues(getStageResults(design, dataInput = data))

## End(Not run)
```

`getSampleSizeCounts` *Get Sample Size Counts*

Description

Returns the sample size for testing the ratio of mean rates of negative binomial distributed event numbers in two samples at given effect.

Usage

```
getSampleSizeCounts(
  design = NULL,
  ...,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  lambda = NA_real_,
  theta = NA_real_,
  thetaH0 = 1,
  overdispersion = 0,
  fixedExposureTime = NA_real_,
  accrualTime = NA_real_,
  accrualIntensity = NA_real_,
  followUpTime = NA_real_,
  maxNumberOfSubjects = NA_integer_,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`lambda1`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the active treatment group, there is no default.
`lambda2`	A numeric value that represents the assumed rate of a homogeneous Poisson process in the control group, there is no default.
`lambda`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the pooled treatment groups, there is no default.
`theta`	A numeric value or vector that represents the assumed mean ratios lambda1/lambda2 of a homogeneous Poisson process, there is no default.
`thetaH0`	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`overdispersion` A numeric value that represents the assumed overdispersion of the negative binomial distribution, default is `0`.

`fixedExposureTime`
If specified, the fixed time of exposure per subject for count data, there is no default.

`accrualTime` If specified, the assumed accrual time interval(s) for the study, there is no default.

`accrualIntensity`
If specified, the assumed accrual intensities for the study, there is no default.

`followUpTime` If specified, the assumed (additional) follow-up time for the study, there is no default. The total study duration is `accrualTime + followUpTime`.

`maxNumberOfSubjects`
`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. If `allocationRatioPlanned = 0` is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.

Details

At given design the function calculates the information, and stage-wise and maximum sample size for testing mean rates of negative binomial distributed event numbers in two samples at given effect. The sample size calculation is performed either for a fixed exposure time or a variable exposure time with fixed follow-up. For the variable exposure time case, at given maximum sample size the necessary follow-up time is calculated. The planned calendar time of interim stages is calculated if an accrual time is defined. Additionally, an allocation ratio = `n1 / n2` can be specified where `n1` and `n2` are the number of subjects in the two treatment groups. A null hypothesis value `thetaH0` can also be specified.

Value

Returns a `TrialDesignPlan` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other sample size functions: getSampleSizeMeans(), getSampleSizeRates(), getSampleSizeSurvival()

Examples

```
## Not run:
# Fixed sample size trial where a therapy is assumed to decrease the
# exacerbation rate from 1.4 to 1.05 (25% decrease) within an observation
# period of 1 year, i.e., each subject has an equal follow-up of 1 year.
# The sample size that yields 90% power at significance level 0.025 for
# detecting such a difference, if the overdispersion is assumed to be
# equal to 0.5, is obtained by
getSampleSizeCounts(alpha = 0.025, beta = 0.1, lambda2 = 1.4,
    theta = 0.75, overdispersion = 0.5, fixedExposureTime = 1)

# Noninferiority test with blinded sample size reassessment to reproduce
# Table 2 from Friede and Schmidli (2010):
getSampleSizeCounts(alpha = 0.025, beta = 0.2, lambda = 1, theta = 1,
    thetaH0 = 1.15, overdispersion = 0.4, fixedExposureTime = 1)

# Group sequential alpha and beta spending function design with O'Brien and
# Fleming type boundaries: Estimate observation time under uniform
# recruitment of patients over 6 months and a fixed exposure time of 12
# months (lambda1, lambda2, and overdispersion as specified):
getSampleSizeCounts(design = getDesignGroupSequential(
        kMax = 3, alpha = 0.025, beta = 0.2,
        typeOfDesign = "asOF", typeBetaSpending = "bsOF"),
    lambda1 = 0.2, lambda2 = 0.3, overdispersion = 1,
    fixedExposureTime = 12, accrualTime = 6)

# Group sequential alpha spending function design with O'Brien and Fleming
# type boundaries: Sample size for variable exposure time with uniform
# recruitment over 1.25 months and study time (accrual + followup) = 4
# (lambda1, lambda2, and overdispersion as specified, no futility stopping):
getSampleSizeCounts(design = getDesignGroupSequential(
        kMax = 3, alpha = 0.025, beta = 0.2, typeOfDesign = "asOF"),
    lambda1 = 0.0875, lambda2 = 0.125, overdispersion = 5,
    followUpTime = 2.75, accrualTime = 1.25)

## End(Not run)
```

---

getSampleSizeMeans *Get Sample Size Means*

Description

Returns the sample size for testing means in one or two samples.

Usage

```
getSampleSizeMeans(
  design = NULL,
  ...,
  groups = 2L,
  normalApproximation = FALSE,
  meanRatio = FALSE,
  thetaH0 = ifelse(meanRatio, 1, 0),
  alternative = seq(0.2, 1, 0.2),
  stDev = 1,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`groups`	The number of treatment groups (1 or 2), default is `2`.
`normalApproximation`	The type of computation of the p-values. If `TRUE`, the variance is assumed to be known, default is `FALSE`, i.e., the calculations are performed with the t distribution.
`meanRatio`	If `TRUE`, the sample size for one-sided testing of H0: `mu1 / mu2 = thetaH0` is calculated, default is `FALSE`.
`thetaH0`	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified. • *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified. • *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified. • *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified. For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.
`alternative`	The alternative hypothesis value for testing means. This can be a vector of assumed alternatives, default is `seq(0, 1, 0.2)` (power calculations) or `seq(0.2, 1, 0.2)` (sample size calculations).

stDev
: The standard deviation under which the sample size or power calculation is performed, default is 1. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If meanRatio = TRUE is specified, stDev defines the coefficient of variation sigma / mu2.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. If allocationRatioPlanned = 0 is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.

Details

At given design the function calculates the stage-wise and maximum sample size for testing means. In a two treatment groups design, additionally, an allocation ratio = n1 / n2 can be specified where n1 and n2 are the number of subjects in the two treatment groups. A null hypothesis value thetaH0 != 0 for testing the difference of two means or thetaH0 != 1 for testing the ratio of two means can be specified. Critical bounds and stopping for futility bounds are provided at the effect scale (mean, mean difference, or mean ratio, respectively) for each sample size calculation separately.

Value

Returns a TrialDesignPlan object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other sample size functions: getSampleSizeCounts(), getSampleSizeRates(), getSampleSizeSurvival()

Examples

```
## Not run:
# Calculate sample sizes in a fixed sample size parallel group design
# with allocation ratio \code{n1 / n2 = 2} for a range of
# alternative values 1, ..., 5 with assumed standard deviation = 3.5;
# two-sided alpha = 0.05, power 1 - beta = 90%:
getSampleSizeMeans(alpha = 0.05, beta = 0.1, sided = 2, groups = 2,
    alternative = seq(1, 5, 1), stDev = 3.5, allocationRatioPlanned = 2)

# Calculate sample sizes in a three-stage Pocock paired comparison design testing
```

```
# H0: mu = 2 for a range of alternative values 3,4,5 with assumed standard
# deviation = 3.5; one-sided alpha = 0.05, power 1 - beta = 90%:
getSampleSizeMeans(getDesignGroupSequential(typeOfDesign = "P", alpha = 0.05,
    sided = 1, beta = 0.1), groups = 1, thetaH0 = 2,
    alternative = seq(3, 5, 1), stDev = 3.5)

# Calculate sample sizes in a three-stage Pocock two-armed design testing
# H0: mu = 2 for a range of alternative values 3,4,5 with assumed standard
# deviations = 3 and 4, respectively, in the two groups of observations;
# one-sided alpha = 0.05, power 1 - beta = 90%:
getSampleSizeMeans(getDesignGroupSequential(typeOfDesign = "P", alpha = 0.05,
    sided = 1, beta = 0.1), groups = 2,
    alternative = seq(3, 5, 1), stDev = c(3, 4))

## End(Not run)
```

`getSampleSizeRates` *Get Sample Size Rates*

Description

Returns the sample size for testing rates in one or two samples.

Usage

```
getSampleSizeRates(
  design = NULL,
  ...,
  groups = 2L,
  normalApproximation = TRUE,
  conservative = TRUE,
  riskRatio = FALSE,
  thetaH0 = ifelse(riskRatio, 1, 0),
  pi1 = c(0.4, 0.5, 0.6),
  pi2 = 0.2,
  allocationRatioPlanned = NA_real_
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`groups`	The number of treatment groups (1 or 2), default is `2`.
`normalApproximation`	If `FALSE`, the sample size for the case of one treatment group is calculated exactly using the binomial distribution, default is `TRUE`.

`conservative` For the case of one treatment group and `normalApproximation = FALSE`, if `TRUE`, the sample size is calculated such that for larger sample size than the calculated, the power is larger than 1 - beta, for `conservative = FALSE`, the minimum sample size, for which power exceeds 1 - beta is calculated, default is `TRUE`.

`riskRatio` If `TRUE`, the sample size for one-sided testing of H0: `pi1 / pi2 = thetaH0` is calculated, default is `FALSE`.

`thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio).

For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`pi1` A numeric value or vector that represents the assumed probability in the active treatment group if two treatment groups are considered, or the alternative probability for a one treatment group design, default is `seq(0.2, 0.5, 0.1)` (power calculations and simulations) or `seq(0.4, 0.6, 0.1)` (sample size calculations).

`pi2` A numeric value that represents the assumed probability in the reference group if two treatment groups are considered, default is `0.2`.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. If `allocationRatioPlanned = 0` is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.

Details

At given design the function calculates the stage-wise and maximum sample size for testing rates. In a two treatment groups design, additionally, an allocation ratio = `n1 / n2` can be specified where `n1` and `n2` are the number of subjects in the two treatment groups. If a null hypothesis value thetaH0 != 0 for testing the difference of two rates or thetaH0 != 1 for testing the risk ratio is specified, the sample size formula according to Farrington & Manning (Statistics in Medicine, 1990) is used. Critical bounds and stopping for futility bounds are provided at the effect scale (rate, rate difference, or rate ratio, respectively) for each sample size calculation separately. For the two-sample case, the calculation here is performed at fixed pi2 as given as argument in the function.

Value

Returns a `TrialDesignPlan` object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other sample size functions: getSampleSizeCounts(), getSampleSizeMeans(), getSampleSizeSurvival()

Examples

```
## Not run:
# Calculate the stage-wise sample sizes, maximum sample sizes, and the optimum
# allocation ratios for a range of pi1 values when testing
# H0: pi1 - pi2 = -0.1 within a two-stage O'Brien & Fleming design;
# alpha = 0.05 one-sided, power 1 - beta = 90%:
getSampleSizeRates(getDesignGroupSequential(kMax = 2, alpha = 0.05,
    beta = 0.1), groups = 2, thetaH0 = -0.1, pi1 = seq(0.4, 0.55, 0.025),
    pi2 = 0.4, allocationRatioPlanned = 0)

# Calculate the stage-wise sample sizes, maximum sample sizes, and the optimum
# allocation ratios for a range of pi1 values when testing
# H0: pi1 / pi2 = 0.80 within a three-stage O'Brien & Fleming design;
# alpha = 0.025 one-sided, power 1 - beta = 90%:
getSampleSizeRates(getDesignGroupSequential(kMax = 3, alpha = 0.025,
    beta = 0.1), groups = 2, riskRatio = TRUE, thetaH0 = 0.80,
    pi1 = seq(0.3, 0.5, 0.025), pi2 = 0.3, allocationRatioPlanned = 0)

## End(Not run)
```

---

getSampleSizeSurvival *Get Sample Size Survival*

Description

Returns the sample size for testing the hazard ratio in a two treatment groups survival design.

Usage

```
getSampleSizeSurvival(
  design = NULL,
  ...,
  typeOfComputation = c("Schoenfeld", "Freedman", "HsiehFreedman"),
  thetaH0 = 1,
  pi1 = NA_real_,
  pi2 = NA_real_,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  median1 = NA_real_,
  median2 = NA_real_,
  kappa = 1,
  hazardRatio = NA_real_,
  piecewiseSurvivalTime = NA_real_,
  allocationRatioPlanned = NA_real_,
  eventTime = 12,
  accrualTime = c(0, 12),
  accrualIntensity = 0.1,
  accrualIntensityType = c("auto", "absolute", "relative"),
  followUpTime = NA_real_,
  maxNumberOfSubjects = NA_real_,
  dropoutRate1 = 0,
  dropoutRate2 = 0,
  dropoutTime = 12
)
```

Arguments

`design` The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.

`...` Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

`typeOfComputation`
Three options are available: `"Schoenfeld"`, `"Freedman"`, `"HsiehFreedman"`, the default is `"Schoenfeld"`. For details, see Hsieh (Statistics in Medicine, 1992). For non-inferiority testing (i.e., `thetaH0 != 1`), only Schoenfeld's formula can be used.

`thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio).

For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.

- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`pi1` A numeric value or vector that represents the assumed event rate in the treatment group, default is `seq(0.2, 0.5, 0.1)` (power calculations and simulations) or `seq(0.4, 0.6, 0.1)` (sample size calculations).

`pi2` A numeric value that represents the assumed event rate in the control group, default is `0.2`.

`lambda1` The assumed hazard rate in the treatment group, there is no default. `lambda1` can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

`lambda2` The assumed hazard rate in the reference group, there is no default. `lambda2` can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

`median1` The assumed median survival time in the treatment group, there is no default.

`median2` The assumed median survival time in the reference group, there is no default. Must be a positive numeric of length 1.

`kappa` A numeric value > 0. A `kappa != 1` will be used for the specification of the shape of the Weibull distribution. Default is `1`, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only `piecewiselambda` (as a single value) and `kappa` can be specified. This function is equivalent to `pweibull(t, shape = kappa, scale = 1 / lambda)` of the `stats` package, i.e., the scale parameter is `1 / 'hazard rate'`.
For example, `getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2)` and `pweibull(q = 130, shape = 4.2, scale = 1 / 0.01)` provide the same result.

`hazardRatio` The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.

`piecewiseSurvivalTime`
A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function
(for details see `getPiecewiseSurvivalTime()`).

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. If `allocationRatioPlanned = 0` is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.

`eventTime` The assumed time under which the event rates are calculated, default is `12`.

`accrualTime` The assumed accrual time intervals for the study, default is `c(0, 12)` (for details see `getAccrualTime()`).

`accrualIntensity`
A numeric vector of accrual intensities, default is the relative intensity `0.1` (for details see `getAccrualTime()`).

`accrualIntensityType`
A character value specifying the accrual intensity input type. Must be one of `"auto"`, `"absolute"`, or `"relative"`; default is `"auto"`, i.e., if all values are < 1 the type is `"relative"`, otherwise it is `"absolute"`.

`followUpTime` The assumed (additional) follow-up time for the study, default is `6`. The total study duration is `accrualTime + followUpTime`.

`maxNumberOfSubjects`
If `maxNumberOfSubjects > 0` is specified, the follow-up time for the required number of events is determined.

`dropoutRate1` The assumed drop-out rate in the treatment group, default is `0`.

`dropoutRate2` The assumed drop-out rate in the control group, default is `0`.

`dropoutTime` The assumed time for drop-out rates in the control and the treatment group, default is `12`.

Details

At given design the function calculates the number of events and an estimate for the necessary number of subjects for testing the hazard ratio in a survival design. It also calculates the time when the required events are expected under the given assumptions (exponentially, piecewise exponentially, or Weibull distributed survival times and constant or non-constant piecewise accrual). Additionally, an allocation ratio = `n1 / n2` can be specified where `n1` and `n2` are the number of subjects in the two treatment groups.

Optional argument `accountForObservationTimes`: if `accountForObservationTimes = TRUE`, the number of subjects is calculated assuming specific accrual and follow-up time, default is `TRUE`.

The formula of Kim & Tsiatis (Biometrics, 1990) is used to calculate the expected number of events under the alternative (see also Lakatos & Lan, Statistics in Medicine, 1992). These formulas are generalized to piecewise survival times and non-constant piecewise accrual over time.

Optional argument `accountForObservationTimes`: if `accountForObservationTimes = FALSE`, only the event rates are used for the calculation of the maximum number of subjects.

Value

Returns a `TrialDesignPlan` object. The following generics (R generic functions) are available for this result object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

Piecewise survival time

The first element of the vector `piecewiseSurvivalTime` must be equal to `0`. `piecewiseSurvivalTime` can also be a list that combines the definition of the time intervals and hazard rates in the reference group. The definition of the survival time in the treatment group is obtained by the specification of the hazard ratio (see examples for details).

Staggered patient entry

accrualTime is the time period of subjects' accrual in a study. It can be a value that defines the end of accrual or a vector. In this case, accrualTime can be used to define a non-constant accrual over time. For this, accrualTime is a vector that defines the accrual intervals. The first element of accrualTime must be equal to 0 and, additionally, accrualIntensity needs to be specified. accrualIntensity itself is a value or a vector (depending on the length of accrualTime) that defines the intensity how subjects enter the trial in the intervals defined through accrualTime.

accrualTime can also be a list that combines the definition of the accrual time and accrual intensity (see below and examples for details).

If the length of accrualTime and the length of accrualIntensity are the same (i.e., the end of accrual is undefined), maxNumberOfSubjects > 0 needs to be specified and the end of accrual is calculated. In that case, accrualIntensity is the number of subjects per time unit, i.e., the absolute accrual intensity.

If the length of accrualTime equals the length of accrualIntensity - 1 (i.e., the end of accrual is defined), maxNumberOfSubjects is calculated if the absolute accrual intensity is given. If all elements in accrualIntensity are smaller than 1, accrualIntensity defines the *relative* intensity how subjects enter the trial. For example, accrualIntensity = c(0.1, 0.2) specifies that in the second accrual interval the intensity is doubled as compared to the first accrual interval. The actual (absolute) accrual intensity is calculated for the calculated or given maxNumberOfSubjects. Note that the default is accrualIntensity = 0.1 meaning that the *absolute* accrual intensity will be calculated.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other sample size functions: getSampleSizeCounts(), getSampleSizeMeans(), getSampleSizeRates()

Examples

```
## Not run:
# Fixed sample size trial with median survival 20 vs. 30 months in treatment and
# reference group, respectively, alpha = 0.05 (two-sided), and power 1 - beta = 90%.
# 20 subjects will be recruited per month up to 400 subjects, i.e., accrual time
# is 20 months.
getSampleSizeSurvival(alpha = 0.05, sided = 2, beta = 0.1, lambda1 = log(2) / 20,
    lambda2 = log(2) / 30, accrualTime = c(0,20), accrualIntensity = 20)

# Fixed sample size with minimum required definitions, pi1 = c(0.4,0.5,0.6) and
# pi2 = 0.2 at event time 12, accrual time 12 and follow-up time 6 as default,
# only alpha = 0.01 is specified
getSampleSizeSurvival(alpha = 0.01)

# Four stage O'Brien & Fleming group sequential design with minimum required
# definitions, pi1 = c(0.4,0.5,0.6) and pi2 = 0.2 at event time 12,
# accrual time 12 and follow-up time 6 as default
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 4))
```

```
# For fixed sample design, determine necessary accrual time if 200 subjects and
# 30 subjects per time unit can be recruited
getSampleSizeSurvival(accrualTime = c(0), accrualIntensity = c(30),
    maxNumberOfSubjects = 200)

# Determine necessary accrual time if 200 subjects and if the first 6 time units
# 20 subjects per time unit can be recruited, then 30 subjects per time unit
getSampleSizeSurvival(accrualTime = c(0, 6), accrualIntensity = c(20, 30),
    maxNumberOfSubjects = 200)

# Determine maximum number of Subjects if the first 6 time units 20 subjects
# per time unit can be recruited, and after 10 time units 30 subjects per time unit
getSampleSizeSurvival(accrualTime = c(0, 6, 10), accrualIntensity = c(20, 30))

# Specify accrual time as a list
at <- list(
    "0 - <6"  = 20,
    "6 - Inf" = 30)
getSampleSizeSurvival(accrualTime = at, maxNumberOfSubjects = 200)

# Specify accrual time as a list, if maximum number of subjects need to be calculated
at <- list(
    "0 - <6"   = 20,
    "6 - <=10" = 30)
getSampleSizeSurvival(accrualTime = at)

# Specify effect size for a two-stage group design with O'Brien & Fleming boundaries
# Effect size is based on event rates at specified event time
# needs to be specified because it should be shown that hazard ratio < 1
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    pi1 = 0.2, pi2 = 0.3, eventTime = 24)

# Effect size is based on event rate at specified event
# time for the reference group and hazard ratio
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    hazardRatio = 0.5, pi2 = 0.3, eventTime = 24)

# Effect size is based on hazard rate for the reference group and hazard ratio
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    hazardRatio = 0.5, lambda2 = 0.02)

# Specification of piecewise exponential survival time and hazard ratios
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    hazardRatio = c(1.5, 1.8, 2))

# Specification of piecewise exponential survival time as a list and hazard ratios
pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04)
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = pws, hazardRatio = c(1.5, 1.8, 2))

# Specification of piecewise exponential survival time for both treatment arms
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
```

```
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    lambda1 = c(0.015, 0.03, 0.06))

# Specification of piecewise exponential survival time as a list
pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04)
getSampleSizeSurvival(design = getDesignGroupSequential(kMax = 2),
    piecewiseSurvivalTime = pws, hazardRatio = c(1.5, 1.8, 2))

# Specify effect size based on median survival times
getSampleSizeSurvival(median1 = 5, median2 = 3)

# Specify effect size based on median survival times of Weibull distribtion with
# kappa = 2
getSampleSizeSurvival(median1 = 5, median2 = 3, kappa = 2)

# Identify minimal and maximal required subjects to
# reach the required events in spite of dropouts
getSampleSizeSurvival(accrualTime = c(0, 18), accrualIntensity = c(20, 30),
    lambda2 = 0.4, lambda1 = 0.3, followUpTime = Inf, dropoutRate1 = 0.001,
    dropoutRate2 = 0.005)
getSampleSizeSurvival(accrualTime = c(0, 18), accrualIntensity = c(20, 30),
    lambda2 = 0.4, lambda1 = 0.3, followUpTime = 0, dropoutRate1 = 0.001,
    dropoutRate2 = 0.005)

## End(Not run)
```

`getSimulationCounts` *Get Simulation Counts*

Description

Returns the simulated power, stopping probabilities, conditional power, and expected sample size for testing mean rates for negative binomial distributed event numbers in the two treatment groups testing situation.

Usage

```
getSimulationCounts(
  design = NULL,
  ...,
  plannedCalendarTime = NA_real_,
  maxNumberOfSubjects = NA_real_,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  lambda = NA_real_,
  theta = NA_real_,
  directionUpper = NA,
  thetaH0 = 1,
  overdispersion = 0,
```

```
  fixedExposureTime = NA_real_,
  accrualTime = NA_real_,
  accrualIntensity = NA_real_,
  followUpTime = NA_real_,
  allocationRatioPlanned = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  showStatistics = FALSE
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`plannedCalendarTime`	For simulating count data, the time points where an analysis is planned to be performed. Should be a vector of length `kMax`
`maxNumberOfSubjects`	`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.
`lambda1`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the active treatment group, there is no default.
`lambda2`	A numeric value that represents the assumed rate of a homogeneous Poisson process in the control group, there is no default.
`lambda`	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the pooled treatment groups, there is no default.
`theta`	A numeric value or vector that represents the assumed mean ratios lambda1/lambda2 of a homogeneous Poisson process, there is no default.
`directionUpper`	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.
`thetaH0`	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified. • *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified. • *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified. • *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`overdispersion` A numeric value that represents the assumed overdispersion of the negative binomial distribution, default is `0`.

`fixedExposureTime`
If specified, the fixed time of exposure per subject for count data, there is no default.

`accrualTime` If specified, the assumed accrual time interval(s) for the study, there is no default.

`accrualIntensity`
If specified, the assumed accrual intensities for the study, there is no default.

`followUpTime` If specified, the assumed (additional) follow-up time for the study, there is no default. The total study duration is `accrualTime + followUpTime`.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

`maxNumberOfIterations`
The number of simulation iterations, default is `1000`. Must be a positive integer of length 1.

`seed` The seed to reproduce the simulation, default is a random seed.

`showStatistics` Logical. If `TRUE`, summary statistics of the simulated data are displayed for the `print` command, otherwise the output is suppressed, default is `FALSE`.

Details

At given design the function simulates the power, stopping probabilities, conditional power, and expected sample size at given number of subjects and parameter configuration. Additionally, an allocation `ratio = n1/n2` and a null hypothesis value `thetaH0` can be specified.

Value

Returns a `SimulationResults` object. The following generics (R generic functions) are available for this object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

Simulation Data

The summary statistics "Simulated data" contains the following parameters: median range; mean +/-sd

`$show(showStatistics = FALSE)` or `$setShowStatistics(FALSE)` can be used to disable the output of the aggregated simulated data.

`getData()` can be used to get the aggregated simulated data from the object as `data.frame`. The data frame contains the following columns:

1. `iterationNumber`: The number of the simulation iteration.
2. `stageNumber`: The stage.
3. `lambda1`: The assumed or derived event rate in the treatment group.
4. `lambda2`: The assumed or derived event rate in the control group.
5. `accrualTime`: The assumed accrualTime.
6. `followUpTime`: The assumed followUpTime.
7. `overdispersion`: The assumed overdispersion.
8. `fixedFollowUp`: The assumed fixedFollowUp.
9. `numberOfSubjects`: The number of subjects under consideration when the (interim) analysis takes place.
10. `rejectPerStage`: 1 if null hypothesis can be rejected, 0 otherwise.
11. `futilityPerStage`: 1 if study should be stopped for futility, 0 otherwise.
12. `testStatistic`: The test statistic that is used for the test decision
13. `estimatedLambda1`: The estimated rate in treatment group 1.
14. `estimatedLambda2`: The estimated rate in treatment group 2.
15. `estimatedOverdispersion`: The estimated overdispersion.
16. `infoAnalysis`: The Fisher information at interim stage.
17. `trialStop`: `TRUE` if study should be stopped for efficacy or futility or final stage, `FALSE` otherwise.
18. `conditionalPowerAchieved`: Not yet available

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

Examples

```
## Not run:
# Fixed sample size design with two groups, fixed exposure time
getSimulationCounts(
    theta = 1.8,
    lambda2 = 0.2,
    maxNumberOfSubjects = 200,
```

```
    plannedCalendarTime = 8,
    maxNumberOfIterations = 1000,
    fixedExposureTime = 6,
    accrualTime = 3,
    overdispersion = 2)

# Group sequential design alpha spending function design with O'Brien and
# Fleming type boundaries: Power and test characteristics for N = 264,
# under variable exposure time with uniform recruitment over 1.25 months,
# study time (accrual + followup) = 4, interim analysis take place after
# equidistant time points (lambda1, lambda2, and overdispersion as specified,
# no futility stopping):
dOF <- getDesignGroupSequential(
    kMax = 3,
    alpha = 0.025,
    beta = 0.2,
    typeOfDesign = "asOF")

getSimulationCounts(design = dOF,
    lambda1 = seq(0.04, 0.12, 0.02),
    lambda2 = 0.12,
    directionUpper = FALSE,
    overdispersion = 5,
    plannedCalendarTime = (1:3)/3*4,
    maxNumberOfSubjects = 264,
    followUpTime = 2.75,
    accrualTime = 1.25,
    maxNumberOfIterations = 1000)

## End(Not run)
```

`getSimulationEnrichmentMeans` *Get Simulation Enrichment Means*

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size or testing means in an enrichment design testing situation.

Usage

```
getSimulationEnrichmentMeans(
  design = NULL,
  ...,
  effectList = NULL,
  intersectionTest = c("Simes", "SpiessensDebois", "Bonferroni", "Sidak"),
  stratifiedAnalysis = TRUE,
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  epsilonValue = NA_real_,
```

```
  rValue = NA_real_,
  threshold = -Inf,
  plannedSubjects = NA_integer_,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  stDevH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  selectPopulationsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
effectList	List of subsets, prevalences, and effect sizes with columns and number of rows reflecting the different situations to consider (see examples).
intersectionTest	Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Four options are available in enrichment designs: "SpiessensDebois", "Bonferroni", "Simes", and "Sidak", default is "Simes".
stratifiedAnalysis	Logical. For enrichment designs, typically a stratified analysis should be chosen. For testing rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is TRUE.
adaptations	A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).
typeOfSelection	The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", default is "best". For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.
effectMeasure	Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".
successCriterion	Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected

treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedSubjects
: plannedSubjects is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, plannedSubjects refers to the number of subjects per selected active arm.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfSubjectsPerStage with length kMax determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs minNumberOfSubjectsPerStage refers to the minimum number of subjects per selected active arm.

maxNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfSubjectsPerStage with length kMax determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs maxNumberOfSubjectsPerStage refers to the maximum number of subjects per selected active arm.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1 — If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

stDevH1 — If specified, the value of the standard deviation under which the conditional power or sample size recalculation calculation is performed, default is the value of stDev.

maxNumberOfIterations — The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed — The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction — Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt (see details and examples).

selectPopulationsFunction — Optionally, a function can be entered that defines the way of how populations are selected. This function is allowed to depend on effectVector with length populations stage, conditionalPower, conditionalCriticalValue, plannedSubjects/planne allocationRatioPlanned, selectedPopulations, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics — Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected sample size at given number of subjects, parameter configuration, and population selection rule in the enrichment situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment groups as compared to the control group.

The definition of thetaH1 and/or stDevH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfSubjectsPerStage, and maxNumberOfSubjectsPerStage (or calcSubjectsFunction) are defined.

calcSubjectsFunction
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables stage, selectedPopulations, plannedSubjects, allocationRatioPlanned, minNumberOfSubjectsPerStage, maxNumberOfSubjectsPerStage, conditionalPower, conditionalCriticalValue, overallEffects, and stDevH1. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,

- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Assess a population selection strategy with one subset population.
# If the subset is better than the full population, then the subset
# is selected for the second stage, otherwise the full. Print and plot
# design characteristics.

# Define design
designIN <- getDesignInverseNormal(kMax = 2)

# Define subgroups and their prevalences
subGroups <- c("S", "R") # fixed names!
prevalences <- c(0.2, 0.8)

# Define effect matrix and variability
effectR <- 0.2
m <- c()
for (effectS in seq(0, 0.5, 0.25)) {
    m <- c(m, effectS, effectR)
}
effects <- matrix(m, byrow = TRUE, ncol = 2)
stDev <- c(0.4, 0.8)

# Define effect list
effectList <- list(subGroups=subGroups, prevalences=prevalences, stDevs = stDev, effects = effects)

# Perform simulation
simResultsPE <- getSimulationEnrichmentMeans(design = designIN,
    effectList = effectList, plannedSubjects = c(50, 100),
    maxNumberOfIterations = 100)
print(simResultsPE)

# Assess the design characteristics of a user defined selection
# strategy in a three-stage design with no interim efficacy stop
# using the inverse normal method for combining the stages.
# Only the second interim is used for a selecting of a study
# population. There is a small probability for stopping the trial
# at the first interim.

# Define design
designIN2 <- getDesignInverseNormal(typeOfDesign = "noEarlyEfficacy", kMax = 3)

# Define selection function
```

```
mySelection <- function(effectVector, stage) {
    selectedPopulations <- rep(TRUE, 3)
    if (stage == 2) {
        selectedPopulations <- (effectVector >= c(1, 2, 3))
    }
    return(selectedPopulations)
}

# Define subgroups and their prevalences
subGroups <- c("S1", "S12", "S2", "R")   # fixed names!
prevalences <- c(0.2, 0.3, 0.4, 0.1)

effectR <- 1.5
effectS12 = 5
m <- c()
for (effectS1 in seq(0, 5, 1)) {
    for (effectS2 in seq(0, 5, 1)) {
        m <- c(m, effectS1, effectS12, effectS2, effectR)
    }
}
effects <- matrix(m, byrow = TRUE, ncol = 4)
stDev <- 10

# Define effect list
effectList <- list(subGroups=subGroups, prevalences=prevalences, stDevs = stDev, effects = effects)

# Perform simulation
simResultsPE <- getSimulationEnrichmentMeans(
    design = designIN2,
    effectList = effectList,
    typeOfSelection = "userDefined",
    selectPopulationsFunction = mySelection,
    intersectionTest = "Simes",
    plannedSubjects = c(50, 100, 150),
    maxNumberOfIterations = 100)
print(simResultsPE)
if (require(ggplot2)) plot(simResultsPE, type = 3)

## End(Not run)
```

`getSimulationEnrichmentRates` *Get Simulation Enrichment Rates*

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size for testing rates in an enrichment design testing situation.

Usage

```
getSimulationEnrichmentRates(
  design = NULL,
```

```
  ...,
  effectList = NULL,
  intersectionTest = c("Simes", "SpiessensDebois", "Bonferroni", "Sidak"),
  stratifiedAnalysis = TRUE,
  directionUpper = NA,
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  epsilonValue = NA_real_,
  rValue = NA_real_,
  threshold = -Inf,
  plannedSubjects = NA_real_,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  piTreatmentH1 = NA_real_,
  piControlH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  selectPopulationsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
effectList	List of subsets, prevalences, and effect sizes with columns and number of rows reflecting the different situations to consider (see examples).
intersectionTest	Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Four options are available in enrichment designs: "SpiessensDebois", "Bonferroni", "Simes", and "Sidak", default is "Simes".
stratifiedAnalysis	Logical. For enrichment designs, typically a stratified analysis should be chosen. For testing rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is TRUE.
directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.
adaptations	A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).
typeOfSelection	The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", de-

fault is "best".
For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.

effectMeasure
: Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".

successCriterion
: Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedSubjects
: plannedSubjects is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, plannedSubjects refers to the number of subjects per selected active arm.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfSubjectsPerStage with length kMax determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs minNumberOfSubjectsPerStage refers to the minimum number of subjects per selected active arm.

maxNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfSubjectsPerStage with length kMax determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both

treatment arms. For multi-arm designs maxNumberOfSubjectsPerStage refers to the maximum number of subjects per selected active arm.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

piTreatmentH1
: If specified, the assumed probabilities in the active arm under which the sample size recalculation was performed and the conditional power was calculated.

piControlH1
: If specified, the assumed probabilities in the control arm under which the sample size recalculation was performed and the conditional power was calculated.

maxNumberOfIterations
: The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed
: The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt (see details and examples).

selectPopulationsFunction
: Optionally, a function can be entered that defines the way of how populations are selected. This function is allowed to depend on effectVector with length populations stage, conditionalPower, conditionalCriticalValue, plannedSubjects/planne allocationRatioPlanned, selectedPopulations, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected sample size at given number of subjects, parameter configuration, and treatment arm selection rule in the enrichment situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment groups as compared to the control group.

The definition of piTreatmentH1 and/or piControlH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfSubjectsPerStage, and maxNumberOfSubjectsPerStage (or calcSubjectsFunction) are defined.

calcSubjectsFunction
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables stage, selectedPopulations, directionUpper, plannedSubjects, allocationRatioPlanned, minNumberOfSubjectsPerStage, maxNumberOfSubjectsPerStage, conditionalPower, conditionalCriticalValue, overallRatesTreatment,

overallRatesControl, piTreatmentH1, and piControlH1. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Assess a population selection strategy with two subset populations and
# a binary endpoint using a stratified analysis. No early efficacy stop,
# weighted inverse normal method with weight sqrt(0.4).
pi2 <- c(0.3, 0.4, 0.3, 0.55)
pi1Seq <- seq(0.0, 0.2, 0.2)
pi1 <- matrix(rep(pi1Seq, length(pi2)), ncol = length(pi1Seq), byrow = TRUE) + pi2
effectList <- list(
    subGroups = c("S1", "S2", "S12", "R"),
    prevalences = c(0.1, 0.4, 0.2, 0.3),
    piControl = pi2,
    piTreatments = expand.grid(pi1[1, ], pi1[2, ], pi1[3, ], pi1[4, ])
)
design <- getDesignInverseNormal(informationRates = c(0.4, 1),
    typeOfDesign = "noEarlyEfficacy")
simResultsPE <- getSimulationEnrichmentRates(design,
    plannedSubjects = c(150, 300),
    allocationRatioPlanned = 1.5, directionUpper = TRUE,
    effectList = effectList, stratifiedAnalysis = TRUE,
    intersectionTest = "Sidak",
    typeOfSelection = "epsilon", epsilonValue = 0.025,
    maxNumberOfIterations = 100)
print(simResultsPE)

## End(Not run)
```

`getSimulationEnrichmentSurvival` *Get Simulation Enrichment Survival*

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size for testing hazard ratios in an enrichment design testing situation. In contrast to `getSimulationSurvival()` (where survival times are simulated), normally distributed logrank test statistics are simulated.

Usage

```
getSimulationEnrichmentSurvival(
  design = NULL,
  ...,
  effectList = NULL,
  intersectionTest = c("Simes", "SpiessensDebois", "Bonferroni", "Sidak"),
  stratifiedAnalysis = TRUE,
  directionUpper = NA,
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  epsilonValue = NA_real_,
  rValue = NA_real_,
  threshold = -Inf,
  plannedEvents = NA_real_,
  allocationRatioPlanned = NA_real_,
  minNumberOfEventsPerStage = NA_real_,
  maxNumberOfEventsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcEventsFunction = NULL,
  selectPopulationsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

`design`	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`effectList`	List of subsets, prevalences, and effect sizes with columns and number of rows reflecting the different situations to consider (see examples).

intersectionTest
: Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Four options are available in enrichment designs: "SpiessensDebois", "Bonferroni", "Simes", and "Sidak", default is "Simes".

stratifiedAnalysis
: Logical. For enrichment designs, typically a stratified analysis should be chosen. For testing rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is TRUE.

directionUpper
: Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.

adaptations
: A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).

typeOfSelection
: The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", default is "best".
For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.

effectMeasure
: Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".

successCriterion
: Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedEvents
: plannedEvents is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, it is the number of events for both treatment arms. For multi-arm designs, plannedEvents refers to the overall number of events for the selected arms plus control.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length

kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfEventsPerStage with length kMax determines the minimum number of events per stage (i.e., not cumulated), the first element is not taken into account.

maxNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfEventsPerStage with length kMax determines the maximum number of events per stage (i.e., not cumulated), the first element is not taken into account.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1
: If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

maxNumberOfIterations
: The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed
: The seed to reproduce the simulation, default is a random seed.

calcEventsFunction
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, event number recalculation is performed with conditional power and specified minNumberOfEventsPerStage and maxNumberOfEventsPerStage (see details and examples).

selectPopulationsFunction
: Optionally, a function can be entered that defines the way of how populations are selected. This function is allowed to depend on effectVector with length populations stage, conditionalPower, conditionalCriticalValue, plannedSubjects/plann allocationRatioPlanned, selectedPopulations, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected event number at given number of events, parameter configuration, and population

selection rule in the enrichment situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment group as compared to the control group.

The definition of thetaH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfEventsPerStage, and maxNumberOfEventsPerStage (or calcEventsFunction) are defined.

calcEventsFunction
This function returns the number of events at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables stage, selectedPopulations, plannedEvents, directionUpper, allocationRatioPlanned, minNumberOfEventsPerStage, maxNumberOfEventsP conditionalPower, conditionalCriticalValue, and overallEffects. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Assess a population selection strategy with one subset population and
# a survival endpoint. The considered situations are defined through the
# event rates yielding a range of hazard ratios in the subsets. Design
# with O'Brien and Fleming alpha spending and a reassessment of event
# number in the first interim based on conditional power and assumed
# hazard ratio using weighted inverse normal combination test.

subGroups <- c("S", "R")
prevalences <- c(0.40, 0.60)

p2 <- c(0.3, 0.4)
range1 <- p2[1] + seq(0, 0.3, 0.05)

p1 <- c()
for (x1 in range1) {
    p1 <- c(p1, x1, p2[2] + 0.1)
}
hazardRatios <- log(matrix(1 - p1, byrow = TRUE, ncol = 2)) /
    matrix(log(1 - p2), byrow = TRUE, ncol = 2,
```

```
    nrow = length(range1))

effectList <- list(subGroups=subGroups, prevalences=prevalences,
    hazardRatios = hazardRatios)

design <- getDesignInverseNormal(informationRates = c(0.3, 0.7, 1),
    typeOfDesign = "asOF")

simResultsPE <- getSimulationEnrichmentSurvival(design,
    plannedEvents = c(40, 90, 120),
    effectList = effectList,
    typeOfSelection = "rbest", rValue = 2,
    conditionalPower = 0.8, minNumberOfEventsPerStage = c(NA, 50, 30),
    maxNumberOfEventsPerStage = c(NA, 150, 30), thetaH1 = 4 / 3,
    maxNumberOfIterations = 100)
print(simResultsPE)

## End(Not run)
```

`getSimulationMeans` *Get Simulation Means*

Description

Returns the simulated power, stopping probabilities, conditional power, and expected sample size for testing means in a one or two treatment groups testing situation.

Usage

```
getSimulationMeans(
  design = NULL,
  ...,
  groups = 2L,
  normalApproximation = TRUE,
  meanRatio = FALSE,
  thetaH0 = ifelse(meanRatio, 1, 0),
  alternative = seq(0, 1, 0.2),
  stDev = 1,
  plannedSubjects = NA_real_,
  directionUpper = NA,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  stDevH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

`design` The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate `alpha`, Type II error rate `beta`, `twoSidedPower`, and `sided` can be directly entered as argument where necessary.

`...` Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

`groups` The number of treatment groups (1 or 2), default is `2`.

`normalApproximation` The type of computation of the p-values. Default is `TRUE`, i.e., normally distributed test statistics are generated. If `FALSE`, the t test is used for calculating the p-values, i.e., t distributed test statistics are generated.

`meanRatio` If `TRUE`, the design characteristics for one-sided testing of H0: `mu1 / mu2 = thetaH0` are simulated, default is `FALSE`.

`thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio).

For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`alternative` The alternative hypothesis value for testing means under which the data is simulated. This can be a vector of assumed alternatives, default is `seq(0, 1, 0.2)`.

`stDev` The standard deviation under which the data is simulated, default is `1`. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If `meanRatio = TRUE` is specified, `stDev` defines the coefficient of variation `sigma / mu2`.

`plannedSubjects` `plannedSubjects` is a numeric vector of length `kMax` (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, `plannedSubjects` refers to the number of subjects per selected active arm.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`allocationRatioPlanned` The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it

can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfSubjectsPerStage with length kMax determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs minNumberOfSubjectsPerStage refers to the minimum number of subjects per selected active arm.

maxNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfSubjectsPerStage with length kMax determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs maxNumberOfSubjectsPerStage refers to the maximum number of subjects per selected active arm.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1
: If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

stDevH1
: If specified, the value of the standard deviation under which the conditional power or sample size recalculation calculation is performed, default is the value of stDev.

maxNumberOfIterations
: The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed
: The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt (see details and examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, conditional power, and expected sample size at given number of subjects and parameter configuration. Additionally, an

allocation ratio = n1/n2 can be specified where n1 and n2 are the number of subjects in the two treatment groups.

The definition of thetaH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfSubjectsPerStage, and maxNumberOfSubjectsPerStage (or calcSubjectsFunction) are defined.

calcSubjectsFunction
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on variables stage, meanRatio, thetaH0, groups, plannedSubjects, sampleSizesPerStage, directionUpper, allocationRatioPlanned, minNumberOfSubjectsPerStage, maxNumberOfSubjectsPerStage, conditionalPower, conditionalCriticalValu thetaH1, and stDevH1. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Simulation Data

The summary statistics "Simulated data" contains the following parameters: median range; mean +/-sd

$show(showStatistics = FALSE) or $setShowStatistics(FALSE) can be used to disable the output of the aggregated simulated data.

Example 1:
```
simulationResults <- getSimulationMeans(plannedSubjects = 40)
simulationResults$show(showStatistics = FALSE)
```

Example 2:
```
simulationResults <- getSimulationMeans(plannedSubjects = 40)
simulationResults$setShowStatistics(FALSE)
simulationResults
```

getData() can be used to get the aggregated simulated data from the object as data.frame. The data frame contains the following columns:

1. iterationNumber: The number of the simulation iteration.
2. stageNumber: The stage.
3. alternative: The alternative hypothesis value.
4. numberOfSubjects: The number of subjects under consideration when the (interim) analysis takes place.
5. rejectPerStage: 1 if null hypothesis can be rejected, 0 otherwise.

6. `futilityPerStage`: 1 if study should be stopped for futility, 0 otherwise.
7. `testStatistic`: The test statistic that is used for the test decision, depends on which design was chosen (group sequential, inverse normal, or Fisher's combination test).
8. `testStatisticsPerStage`: The test statistic for each stage if only data from the considered stage is taken into account.
9. `effectEstimate`: Overall simulated standardized effect estimate.
10. `trialStop`: `TRUE` if study should be stopped for efficacy or futility or final stage, `FALSE` otherwise.
11. `conditionalPowerAchieved`: The conditional power for the subsequent stage of the trial for selected sample size and effect. The effect is either estimated from the data or can be user defined with `thetaH1`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

Examples

```
## Not run:
# Fixed sample size design with two groups, total sample size 40,
# alternative = c(0, 0.2, 0.4, 0.8, 1), and standard deviation = 1 (the default)
getSimulationMeans(plannedSubjects = 40, maxNumberOfIterations = 10)

# Increase number of simulation iterations and compare results
# with power calculator using normal approximation
getSimulationMeans(
    alternative = 0:4, stDev = 5,
    plannedSubjects = 40, maxNumberOfIterations = 1000
)
getPowerMeans(
    alternative = 0:4, stDev = 5,
    maxNumberOfSubjects = 40, normalApproximation = TRUE
)

# Do the same for a three-stage O'Brien&Fleming inverse
# normal group sequential design with non-binding futility stops
designIN <- getDesignInverseNormal(typeOfDesign = "OF", futilityBounds = c(0, 0))
x <- getSimulationMeans(designIN,
    alternative = c(0:4), stDev = 5,
    plannedSubjects = c(20, 40, 60), maxNumberOfIterations = 1000
)
getPowerMeans(designIN,
    alternative = 0:4, stDev = 5,
    maxNumberOfSubjects = 60, normalApproximation = TRUE
)

# Assess power and average sample size if a sample size increase is foreseen
# at conditional power 80% for each subsequent stage based on observed overall
# effect and specified minNumberOfSubjectsPerStage and
# maxNumberOfSubjectsPerStage
```

```
getSimulationMeans(designIN,
    alternative = 0:4, stDev = 5,
    plannedSubjects = c(20, 40, 60),
    minNumberOfSubjectsPerStage = c(NA, 20, 20),
    maxNumberOfSubjectsPerStage = c(NA, 80, 80),
    conditionalPower = 0.8,
    maxNumberOfIterations = 50
)

# Do the same under the assumption that a sample size increase only takes
# place at the first interim. The sample size for the third stage is set equal
# to the second stage sample size.
mySampleSizeCalculationFunction <- function(..., stage,
        minNumberOfSubjectsPerStage,
        maxNumberOfSubjectsPerStage,
        sampleSizesPerStage,
        conditionalPower,
        conditionalCriticalValue,
        allocationRatioPlanned,
        thetaH1,
        stDevH1) {
    if (stage <= 2) {
        # Note that allocationRatioPlanned is as a vector of length kMax
        stageSubjects <- (1 + allocationRatioPlanned[stage])^2 /
            allocationRatioPlanned[stage] *
            (max(0, conditionalCriticalValue + stats::qnorm(conditionalPower)))^2 /
            (max(1e-12, thetaH1 / stDevH1))^2
        stageSubjects <- min(max(
            minNumberOfSubjectsPerStage[stage],
            stageSubjects
        ), maxNumberOfSubjectsPerStage[stage])
    } else {
        stageSubjects <- sampleSizesPerStage[stage - 1]
    }
    return(stageSubjects)
}
getSimulationMeans(designIN,
    alternative = 0:4, stDev = 5,
    plannedSubjects = c(20, 40, 60),
    minNumberOfSubjectsPerStage = c(NA, 20, 20),
    maxNumberOfSubjectsPerStage = c(NA, 80, 80),
    conditionalPower = 0.8,
    calcSubjectsFunction = mySampleSizeCalculationFunction,
    maxNumberOfIterations = 50
)

## End(Not run)
```

`getSimulationMultiArmMeans`

Get Simulation Multi-Arm Means

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size for testing means in a multi-arm treatment groups testing situation.

Usage

```
getSimulationMultiArmMeans(
  design = NULL,
  ...,
  activeArms = NA_integer_,
  effectMatrix = NULL,
  typeOfShape = c("linear", "sigmoidEmax", "userDefined"),
  muMaxVector = seq(0, 1, 0.2),
  gED50 = NA_real_,
  slope = 1,
  doseLevels = NA_real_,
 intersectionTest = c("Dunnett", "Bonferroni", "Simes", "Sidak", "Hierarchical"),
  stDev = 1,
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  epsilonValue = NA_real_,
  rValue = NA_real_,
  threshold = -Inf,
  plannedSubjects = NA_integer_,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  stDevH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  selectArmsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
activeArms	The number of active treatment arms to be compared with control, default is 3.
effectMatrix	Matrix of effect sizes with activeArms columns and number of rows reflecting the different situations to consider.
typeOfShape	The shape of the dose-response relationship over the treatment groups. This can be either "linear", "sigmoidEmax", or "userDefined", default is "linear".

For "linear", muMaxVector specifies the range of effect sizes for the treatment group with highest response. If "sigmoidEmax" is selected, gED50 and slope has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For "sigmoidEmax", muMaxVector specifies the range of effect sizes for the treatment group with response according to infinite dose. If "userDefined" is selected, effectMatrix has to be entered.

muMaxVector
: Range of effect sizes for the treatment group with highest response for "linear" and "sigmoidEmax" model, default is seq(0, 1, 0.2).

gED50
: If typeOfShape = "sigmoidEmax" is selected, gED50 has to be entered to specify the ED50 of the sigmoid Emax model.

slope
: If typeOfShape = "sigmoidEmax" is selected, slope can be entered to specify the slope of the sigmoid Emax model, default is 1.

doseLevels
: The dose levels for the dose response relationship. If not specified, these dose levels are 1,...,activeArms.

intersectionTest
: Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Five options are available in multi-arm designs: "Dunnett", "Bonferroni", "Simes", "Sidak", and "Hierarchical", default is "Dunnett".

stDev
: The standard deviation under which the data is simulated, default is 1. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If meanRatio = TRUE is specified, stDev defines the coefficient of variation sigma / mu2.

adaptations
: A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).

typeOfSelection
: The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", default is "best".
For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.

effectMeasure
: Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".

successCriterion
: Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedSubjects

plannedSubjects is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, plannedSubjects refers to the number of subjects per selected active arm.

allocationRatioPlanned

The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfSubjectsPerStage

When performing a data driven sample size recalculation, the numeric vector minNumberOfSubjectsPerStage with length kMax determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs minNumberOfSubjectsPerStage refers to the minimum number of subjects per selected active arm.

maxNumberOfSubjectsPerStage

When performing a data driven sample size recalculation, the numeric vector maxNumberOfSubjectsPerStage with length kMax determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs maxNumberOfSubjectsPerStage refers to the maximum number of subjects per selected active arm.

conditionalPower

If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1

If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

stDevH1

If specified, the value of the standard deviation under which the conditional power or sample size recalculation calculation is performed, default is the value of stDev.

maxNumberOfIterations

The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed

The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction

Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with

conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt
(see details and examples).

selectArmsFunction
: Optionally, a function can be entered that defines the way of how treatment arms are selected. This function is allowed to depend on effectVector with length activeArms, stage, conditionalPower, conditionalCriticalValue, plannedSubjects/planne allocationRatioPlanned, selectedArms, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected sample size at given number of subjects, parameter configuration, and treatment arm selection rule in the multi-arm situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment groups as compared to the control group.

The definition of thetaH1 and/or stDevH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfSubjectsPerStage, and maxNumberOfSubjectsPerStage (or calcSubjectsFunction) are defined.

calcSubjectsFunction
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables stage, selectedArms, plannedSubjects, allocationRatioPlanned, minNumberOfSubjectsPerStage, maxNumberOfSubjectsPerStage, conditionalPower, conditionalCriticalValue, overallEffects, and stDevH1. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Assess a treatment-arm selection strategy with three active arms,
# if the better of the arms is selected for the second stage, and
# compare it with the no-selection case.
# Assume a linear dose-response relationship
maxNumberOfIterations <- 100
designIN <- getDesignInverseNormal(typeOfDesign = "OF", kMax = 2)
sim <- getSimulationMultiArmMeans(design = designIN,
    activeArms = 3, typeOfShape = "linear",
    muMaxVector = seq(0,0.8,0.2),
    intersectionTest = "Simes",
    typeOfSelection = "best",
    plannedSubjects = c(30,60),
    maxNumberOfIterations = maxNumberOfIterations)

sim0 <- getSimulationMultiArmMeans(design = designIN,
    activeArms = 3, typeOfShape = "linear",
    muMaxVector = seq(0,0.8,0.2),
    intersectionTest = "Simes",
    typeOfSelection = "all",
    plannedSubjects = c(30,60),
    maxNumberOfIterations = maxNumberOfIterations)

sim$rejectAtLeastOne
sim$expectedNumberOfSubjects

sim0$rejectAtLeastOne
sim0$expectedNumberOfSubjects

# Compare the power of the conditional Dunnett test with the power of the
# combination test using Dunnett's intersection tests if no treatment arm
# selection takes place. Asseume a linear dose-response relationship.
maxNumberOfIterations <- 100
designIN <- getDesignInverseNormal(typeOfDesign = "asUser",
    userAlphaSpending = c(0, 0.025))
designCD <- getDesignConditionalDunnett(secondStageConditioning = TRUE)

index <- 1
for (design in c(designIN, designCD)) {
    results <- getSimulationMultiArmMeans(design, activeArms = 3,
        muMaxVector = seq(0, 1, 0.2), typeOfShape = "linear",
        plannedSubjects = cumsum(rep(20, 2)),
        intersectionTest = "Dunnett",
        typeOfSelection = "all", maxNumberOfIterations = maxNumberOfIterations)
    if (index == 1) {
        drift <- results$effectMatrix[nrow(results$effectMatrix), ]
        plot(drift, results$rejectAtLeastOne, type = "l", lty = 1,
            lwd = 3, col = "black", ylab = "Power")
    } else {
        lines(drift,results$rejectAtLeastOne, type = "l",
            lty = index, lwd = 3, col = "red")
    }
    index <- index + 1
}
legend("topleft", legend=c("Combination Dunnett", "Conditional Dunnett"),
```

```
    col=c("black", "red"), lty = (1:2), cex = 0.8)

# Assess the design characteristics of a user defined selection
# strategy in a two-stage design using the inverse normal method
# with constant bounds. Stopping for futility due to
# de-selection of all treatment arms.
designIN <- getDesignInverseNormal(typeOfDesign = "P", kMax = 2)

mySelection <- function(effectVector) {
    selectedArms <- (effectVector >= c(0, 0.1, 0.3))
    return(selectedArms)
}

results <- getSimulationMultiArmMeans(designIN, activeArms = 3,
    muMaxVector = seq(0, 1, 0.2),
    typeOfShape = "linear",
    plannedSubjects = c(30,60),
    intersectionTest = "Dunnett",
    typeOfSelection = "userDefined",
    selectArmsFunction = mySelection,
    maxNumberOfIterations = 100)

options(rpact.summary.output.size = "medium")
summary(results)
if (require(ggplot2)) plot(results, type = c(5,3,9), grid = 4)

## End(Not run)
```

`getSimulationMultiArmRates`

Get Simulation Multi-Arm Rates

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size for testing rates in a multi-arm treatment groups testing situation.

Usage

```
getSimulationMultiArmRates(
  design = NULL,
  ...,
  activeArms = NA_integer_,
  effectMatrix = NULL,
  typeOfShape = c("linear", "sigmoidEmax", "userDefined"),
  piMaxVector = seq(0.2, 0.5, 0.1),
  piControl = 0.2,
  gED50 = NA_real_,
  slope = 1,
  doseLevels = NA_real_,
 intersectionTest = c("Dunnett", "Bonferroni", "Simes", "Sidak", "Hierarchical"),
  directionUpper = NA,
```

```
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  epsilonValue = NA_real_,
  rValue = NA_real_,
  threshold = -Inf,
  plannedSubjects = NA_real_,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  piTreatmentsH1 = NA_real_,
  piControlH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  selectArmsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
activeArms	The number of active treatment arms to be compared with control, default is 3.
effectMatrix	Matrix of effect sizes with activeArms columns and number of rows reflecting the different situations to consider.
typeOfShape	The shape of the dose-response relationship over the treatment groups. This can be either "linear", "sigmoidEmax", or "userDefined", default is "linear". For "linear", piMaxVector specifies the range of effect sizes for the treatment group with highest response. If "sigmoidEmax" is selected, gED50 and slope has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For "sigmoidEmax", piMaxVector specifies the range of effect sizes for the treatment group with response according to infinite dose. If "userDefined" is selected, effectMatrix has to be entered.
piMaxVector	Range of assumed probabilities for the treatment group with highest response for "linear" and "sigmoidEmax" model, default is seq(0, 1, 0.2).
piControl	If specified, the assumed probability in the control arm for simulation and under which the sample size recalculation is performed.
gED50	If typeOfShape = "sigmoidEmax" is selected, gED50 has to be entered to specify the ED50 of the sigmoid Emax model.
slope	If typeOfShape = "sigmoidEmax" is selected, slope can be entered to specify the slope of the sigmoid Emax model, default is 1.
doseLevels	The dose levels for the dose response relationship. If not specified, these dose levels are 1,...,activeArms.

intersectionTest
: Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Five options are available in multi-arm designs: "Dunnett", "Bonferroni", "Simes", "Sidak", and "Hierarchical", default is "Dunnett".

directionUpper
: Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.

adaptations
: A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).

typeOfSelection
: The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", default is "best".
For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.

effectMeasure
: Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".

successCriterion
: Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedSubjects
: plannedSubjects is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, plannedSubjects refers to the number of subjects per selected active arm.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfSubjectsPerStage with length kMax determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs minNumberOfSubjectsPerStage refers to the minimum number of subjects per selected active arm.

maxNumberOfSubjectsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfSubjectsPerStage with length kMax determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs maxNumberOfSubjectsPerStage refers to the maximum number of subjects per selected active arm.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

piTreatmentsH1
: If specified, the assumed probability in the active treatment arm(s) under which the sample size recalculation is performed.

piControlH1
: If specified, the assumed probability in the reference group (if different from piControl) for which the conditional power was calculated.

maxNumberOfIterations
: The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed
: The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt (see details and examples).

selectArmsFunction
: Optionally, a function can be entered that defines the way of how treatment arms are selected. This function is allowed to depend on effectVector with length activeArms, stage, conditionalPower, conditionalCriticalValue, plannedSubjects/planne allocationRatioPlanned, selectedArms, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected sample size at given number of subjects, parameter configuration, and treatment arm

selection rule in the multi-arm situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment groups as compared to the control group.

The definition of `piTreatmentsH1` and/or `piControlH1` makes only sense if `kMax` > 1 and if `conditionalPower`, `minNumberOfSubjectsPerStage`, and `maxNumberOfSubjectsPerStage` (or `calcSubjectsFunction`) are defined.

`calcSubjectsFunction`
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables `stage`, `selectedArms`, `directionUpper`, `plannedSubjects`, `allocationRatioPlanned`, `minNumberOfSubjectsPerStage`, `maxNumberOfSubjectsPerStage`, `conditionalPower`, `conditionalCriticalValue`, `overallRates`, `overallRatesControl`, `piTreatmentsH1`, and `piControlH1`. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a `SimulationResults` object. The following generics (R generic functions) are available for this object:

- `names()` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,
- `plot()` to plot the object,
- `as.data.frame()` to coerce the object to a `data.frame`,
- `as.matrix()` to coerce the object to a `matrix`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

Examples

```
## Not run:
# Simulate the power of the combination test with two interim stages and
# O'Brien & Fleming boundaries using Dunnett's intersection tests if the
# best treatment arm is selected at first interim. Selection only take
# place if a non-negative treatment effect is observed (threshold = 0);
# 20 subjects per stage and treatment arm, simulation is performed for
# four parameter configurations.
design <- getDesignInverseNormal(typeOfDesign = "OF")
effectMatrix <- matrix(c(0.2,0.2,0.2,
    0.4,0.4,0.4,
    0.4,0.5,0.5,
    0.4,0.5,0.6),
    byrow = TRUE, nrow = 4, ncol = 3)
x <- getSimulationMultiArmRates(design = design, typeOfShape = "userDefined",
    effectMatrix = effectMatrix , piControl = 0.2,
    typeOfSelection = "best", threshold = 0, intersectionTest = "Dunnett",
    plannedSubjects = c(20, 40, 60),
    maxNumberOfIterations = 50)
```

```
summary(x)

## End(Not run)
```

`getSimulationMultiArmSurvival` *Get Simulation Multi-Arm Survival*

Description

Returns the simulated power, stopping and selection probabilities, conditional power, and expected sample size for testing hazard ratios in a multi-arm treatment groups testing situation. In contrast to `getSimulationSurvival()` (where survival times are simulated), normally distributed logrank test statistics are simulated.

Usage

```
getSimulationMultiArmSurvival(
  design = NULL,
  ...,
  activeArms = NA_integer_,
  effectMatrix = NULL,
  typeOfShape = c("linear", "sigmoidEmax", "userDefined"),
  omegaMaxVector = seq(1, 2.6, 0.4),
  gED50 = NA_real_,
  slope = 1,
  doseLevels = NA_real_,
 intersectionTest = c("Dunnett", "Bonferroni", "Simes", "Sidak", "Hierarchical"),
  directionUpper = NA,
  adaptations = NA,
  typeOfSelection = c("best", "rBest", "epsilon", "all", "userDefined"),
  effectMeasure = c("effectEstimate", "testStatistic"),
  successCriterion = c("all", "atLeastOne"),
  correlationComputation = c("alternative", "null"),
  epsilonValue = NA_real_,
  rValue = NA_real_,
  threshold = -Inf,
  plannedEvents = NA_real_,
  allocationRatioPlanned = NA_real_,
  minNumberOfEventsPerStage = NA_real_,
  maxNumberOfEventsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcEventsFunction = NULL,
  selectArmsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
activeArms	The number of active treatment arms to be compared with control, default is 3.
effectMatrix	Matrix of effect sizes with activeArms columns and number of rows reflecting the different situations to consider.
typeOfShape	The shape of the dose-response relationship over the treatment groups. This can be either "linear", "sigmoidEmax", or "userDefined", default is "linear". For "linear", omegaMaxVector specifies the range of effect sizes for the treatment group with highest response. If "sigmoidEmax" is selected, gED50 and slope has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For "sigmoidEmax", omegaMaxVector specifies the range of effect sizes for the treatment group with response according to infinite dose. If "userDefined" is selected, effectMatrix has to be entered.
omegaMaxVector	Range of hazard ratios with highest response for "linear" and "sigmoidEmax" model, default is seq(1, 2.6, 0.4).
gED50	If typeOfShape = "sigmoidEmax" is selected, gED50 has to be entered to specify the ED50 of the sigmoid Emax model.
slope	If typeOfShape = "sigmoidEmax" is selected, slope can be entered to specify the slope of the sigmoid Emax model, default is 1.
doseLevels	The dose levels for the dose response relationship. If not specified, these dose levels are 1, ..., activeArms.
intersectionTest	Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Five options are available in multi-arm designs: "Dunnett", "Bonferroni", "Simes", "Sidak", and "Hierarchical", default is "Dunnett".
directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.
adaptations	A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).
typeOfSelection	The way the treatment arms or populations are selected at interim. Five options are available: "best", "rbest", "epsilon", "all", and "userDefined", default is "best". For "rbest" (select the rValue best treatment arms/populations), the parameter rValue has to be specified, for "epsilon" (select treatment arm/population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. If "userDefined" is selected, "selectArmsFunction" or "selectPopulationsFunction" has to be specified.
effectMeasure	Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".
successCriterion	Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected

treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

correlationComputation
: If correlationComputation = "alternative", for simulating log-rank statistics in the many-to-one design, a correlation matrix according to Deng et al. (Biometrics, 2019) accounting for the respective alternative is used; if correlationComputation = "null", a constant correlation matrix valid under the null, i.e., not accounting for the alternative is used, default is "alternative".

epsilonValue
: For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

rValue
: For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

threshold
: Selection criterion: treatment arm / population is selected only if effectMeasure exceeds threshold, default is -Inf. threshold can also be a vector of length activeArms referring to a separate threshold condition over the treatment arms.

plannedEvents
: plannedEvents is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, it is the number of events for both treatment arms. For multi-arm designs, plannedEvents refers to the overall number of events for the selected arms plus control.

allocationRatioPlanned
: The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

minNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfEventsPerStage with length kMax determines the minimum number of events per stage (i.e., not cumulated), the first element is not taken into account.

maxNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfEventsPerStage with length kMax determines the maximum number of events per stage (i.e., not cumulated), the first element is not taken into account.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1 — If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

maxNumberOfIterations — The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed — The seed to reproduce the simulation, default is a random seed.

calcEventsFunction — Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, event number recalculation is performed with conditional power and specified minNumberOfEventsPerStage and maxNumberOfEventsPerStage (see details and examples).

selectArmsFunction — Optionally, a function can be entered that defines the way of how treatment arms are selected. This function is allowed to depend on effectVector with length activeArms, stage, conditionalPower, conditionalCriticalValue, plannedSubjects/planne allocationRatioPlanned, selectedArms, thetaH1 (for means and survival), stDevH1 (for means), overallEffects, and for rates additionally: piTreatmentsH1, piControlH1, overallRates, and overallRatesControl (see examples).

showStatistics — Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, selection probabilities, and expected sample size at given number of subjects, parameter configuration, and treatment arm selection rule in the multi-arm situation. An allocation ratio can be specified referring to the ratio of number of subjects in the active treatment groups as compared to the control group.

The definition of thetaH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfEventsPerStage, and maxNumberOfEventsPerStage (or calcEventsFunction) are defined.

calcEventsFunction
This function returns the number of events at given conditional power and conditional critical value for specified testing situation. The function might depend on the variables stage, selectedArms, plannedEvents, directionUpper, allocationRatioPlanned, minNumberOfEventsPerStage, maxNumberOfEventsP conditionalPower, conditionalCriticalValue, and overallEffects. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Assess different selection rules for a two-stage survival design with
# O'Brien & Fleming alpha spending boundaries and (non-binding) stopping
# for futility if the test statistic is negative.
# Number of events at the second stage is adjusted based on conditional
# power 80% and specified minimum and maximum number of Events.
design <- getDesignInverseNormal(typeOfDesign = "asOF", futilityBounds = 0)

y1 <- getSimulationMultiArmSurvival(design = design, activeArms = 4,
    intersectionTest = "Simes", typeOfShape = "sigmoidEmax",
    omegaMaxVector = seq(1, 2, 0.5), gED50 = 2, slope = 4,
    typeOfSelection = "best", conditionalPower = 0.8,
    minNumberOfEventsPerStage = c(NA_real_, 30),
    maxNumberOfEventsPerStage = c(NA_real_, 90),
    maxNumberOfIterations = 50,
    plannedEvents = c(75, 120))

y2 <- getSimulationMultiArmSurvival(design = design, activeArms = 4,
    intersectionTest = "Simes", typeOfShape = "sigmoidEmax",
    omegaMaxVector = seq(1,2,0.5), gED50 = 2, slope = 4,
    typeOfSelection = "epsilon", epsilonValue = 0.2,
    effectMeasure = "effectEstimate",
    conditionalPower = 0.8, minNumberOfEventsPerStage = c(NA_real_, 30),
    maxNumberOfEventsPerStage = c(NA_real_, 90),
    maxNumberOfIterations = 50,
    plannedEvents = c(75, 120))

y1$effectMatrix

y1$rejectAtLeastOne
y2$rejectAtLeastOne

y1$selectedArms
y2$selectedArms

## End(Not run)
```

---

getSimulationRates *Get Simulation Rates*

Description

Returns the simulated power, stopping probabilities, conditional power, and expected sample size for testing rates in a one or two treatment groups testing situation.

Usage

```
getSimulationRates(
  design = NULL,
  ...,
  groups = 2L,
  normalApproximation = TRUE,
  riskRatio = FALSE,
  thetaH0 = ifelse(riskRatio, 1, 0),
  pi1 = seq(0.2, 0.5, 0.1),
  pi2 = NA_real_,
  plannedSubjects = NA_real_,
  directionUpper = NA,
  allocationRatioPlanned = NA_real_,
  minNumberOfSubjectsPerStage = NA_real_,
  maxNumberOfSubjectsPerStage = NA_real_,
  conditionalPower = NA_real_,
  pi1H1 = NA_real_,
  pi2H1 = NA_real_,
  maxNumberOfIterations = 1000L,
  seed = NA_real_,
  calcSubjectsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
groups	The number of treatment groups (1 or 2), default is 2.
normalApproximation	The type of computation of the p-values. Default is FALSE for testing means (i.e., the t test is used) and TRUE for testing rates and the hazard ratio. For testing rates, if normalApproximation = FALSE is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting normalApproximation = FALSE has no effect.
riskRatio	If TRUE, the design characteristics for one-sided testing of H0: pi1 / pi2 = thetaH0 are simulated, default is FALSE.
thetaH0	The null hypothesis value, default is 0 for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio). For non-inferiority designs, thetaH0 is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value != 0 (or a value != 1 for testing the mean ratio) can be specified. • *rates*: a value != 0 (or a value != 1 for testing the risk ratio pi1 / pi2) can be specified.

- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.
- *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`pi1` A numeric value or vector that represents the assumed probability in the active treatment group if two treatment groups are considered, or the alternative probability for a one treatment group design, default is `seq(0.2, 0.5, 0.1)` (power calculations and simulations) or `seq(0.4, 0.6, 0.1)` (sample size calculations).

`pi2` A numeric value that represents the assumed probability in the reference group if two treatment groups are considered, default is `0.2`.

`plannedSubjects`

`plannedSubjects` is a numeric vector of length `kMax` (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, `plannedSubjects` refers to the number of subjects per selected active arm.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

`allocationRatioPlanned`

The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

`minNumberOfSubjectsPerStage`

When performing a data driven sample size recalculation, the numeric vector `minNumberOfSubjectsPerStage` with length `kMax` determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs `minNumberOfSubjectsPerStage` refers to the minimum number of subjects per selected active arm.

`maxNumberOfSubjectsPerStage`

When performing a data driven sample size recalculation, the numeric vector `maxNumberOfSubjectsPerStage` with length `kMax` determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs `maxNumberOfSubjectsPerStage` refers to the maximum number of subjects per selected active arm.

`conditionalPower`

If `conditionalPower` together with `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPe` (or `minNumberOfEventsPerStage` and `maxNumberOfEventsPerStage` for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent

stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

pi1H1 If specified, the assumed probability in the active treatment group if two treatment groups are considered, or the assumed probability for a one treatment group design, for which the conditional power was calculated.

pi2H1 If specified, the assumed probability in the reference group if two treatment groups are considered, for which the conditional power was calculated.

maxNumberOfIterations
The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

seed The seed to reproduce the simulation, default is a random seed.

calcSubjectsFunction
Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerSt
(see details and examples).

showStatistics Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, conditional power, and expected sample size at given number of subjects and parameter configuration. Additionally, an allocation ratio = n1/n2 can be specified where n1 and n2 are the number of subjects in the two treatment groups.

The definition of pi1H1 and/or pi2H1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfSubjectsPerStage, and maxNumberOfSubjectsPerStage (or calcSubjectsFunction) are defined.

calcSubjectsFunction
This function returns the number of subjects at given conditional power and conditional critical value for specified testing situation. The function might depend on variables stage, riskRatio, thetaH0, groups, plannedSubjects, sampleSizesPerStage, directionUpper, allocationRatioPlanned, minNumberOfSubjectsPerStage, maxNumberOfSubjectsPerStage, conditionalPower, conditionalCriticalValu
overallRate, farringtonManningValue1, and farringtonManningValue2. The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Simulation Data

The summary statistics "Simulated data" contains the following parameters: median range; mean +/-sd

`$show(showStatistics = FALSE)` or `$setShowStatistics(FALSE)` can be used to disable the output of the aggregated simulated data.

Example 1:

```
simulationResults <- getSimulationRates(plannedSubjects = 40)
simulationResults$show(showStatistics = FALSE)
```

Example 2:

```
simulationResults <- getSimulationRates(plannedSubjects = 40)
simulationResults$setShowStatistics(FALSE)
simulationResults
```

`getData()` can be used to get the aggregated simulated data from the object as `data.frame`. The data frame contains the following columns:

1. `iterationNumber`: The number of the simulation iteration.
2. `stageNumber`: The stage.
3. `pi1`: The assumed or derived event rate in the treatment group (if available).
4. `pi2`: The assumed or derived event rate in the control group (if available).
5. `numberOfSubjects`: The number of subjects under consideration when the (interim) analysis takes place.
6. `rejectPerStage`: 1 if null hypothesis can be rejected, 0 otherwise.
7. `futilityPerStage`: 1 if study should be stopped for futility, 0 otherwise.
8. `testStatistic`: The test statistic that is used for the test decision, depends on which design was chosen (group sequential, inverse normal, or Fisher combination test)'
9. `testStatisticsPerStage`: The test statistic for each stage if only data from the considered stage is taken into account.
10. `overallRate1`: The cumulative rate in treatment group 1.
11. `overallRate2`: The cumulative rate in treatment group 2.
12. `stagewiseRates1`: The stage-wise rate in treatment group 1.
13. `stagewiseRates2`: The stage-wise rate in treatment group 2.
14. `sampleSizesPerStage1`: The stage-wise sample size in treatment group 1.
15. `sampleSizesPerStage2`: The stage-wise sample size in treatment group 2.
16. `trialStop`: `TRUE` if study should be stopped for efficacy or futility or final stage, `FALSE` otherwise.
17. `conditionalPowerAchieved`: The conditional power for the subsequent stage of the trial for selected sample size and effect. The effect is either estimated from the data or can be user defined with `pi1H1` and `pi2H1`.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Fixed sample size design (two groups) with total sample
# size 120, pi1 = (0.3,0.4,0.5,0.6) and pi2 = 0.3
getSimulationRates(pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3,
    plannedSubjects = 120, maxNumberOfIterations = 10)

# Increase number of simulation iterations and compare results with power calculator
getSimulationRates(pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3,
    plannedSubjects = 120, maxNumberOfIterations = 50)
getPowerRates(pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, maxNumberOfSubjects = 120)

# Do the same for a two-stage Pocock inverse normal group sequential
# design with non-binding futility stops
designIN <- getDesignInverseNormal(typeOfDesign = "P", futilityBounds = c(0))
getSimulationRates(designIN, pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3,
    plannedSubjects = c(40, 80), maxNumberOfIterations = 50)
getPowerRates(designIN, pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, maxNumberOfSubjects = 80)

# Assess power and average sample size if a sample size reassessment is
# foreseen at conditional power 80% for the subsequent stage (decrease and increase)
# based on observed overall (cumulative) rates and specified minNumberOfSubjectsPerStage
# and maxNumberOfSubjectsPerStage

# Do the same under the assumption that a sample size increase only takes place
# if the rate difference exceeds the value 0.1 at interim. For this, the sample
# size recalculation method needs to be redefined:
mySampleSizeCalculationFunction <- function(..., stage,
        plannedSubjects,
        minNumberOfSubjectsPerStage,
        maxNumberOfSubjectsPerStage,
        conditionalPower,
        conditionalCriticalValue,
        overallRate) {
    if (overallRate[1] - overallRate[2] < 0.1) {
        return(plannedSubjects[stage] - plannedSubjects[stage - 1])
    } else {
        rateUnderH0 <- (overallRate[1] + overallRate[2]) / 2
        stageSubjects <- 2 * (max(0, conditionalCriticalValue *
            sqrt(2 * rateUnderH0 * (1 - rateUnderH0)) +
            stats::qnorm(conditionalPower) * sqrt(overallRate[1] *
            (1 - overallRate[1]) + overallRate[2] * (1 - overallRate[2]))))^2 /
            (max(1e-12, (overallRate[1] - overallRate[2])))^2
        stageSubjects <- ceiling(min(max(
            minNumberOfSubjectsPerStage[stage],
            stageSubjects), maxNumberOfSubjectsPerStage[stage]))
        return(stageSubjects)
    }
```

```
}
getSimulationRates(designIN, pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3,
    plannedSubjects = c(40, 80), minNumberOfSubjectsPerStage = c(40, 20),
    maxNumberOfSubjectsPerStage = c(40, 160), conditionalPower = 0.8,
    calcSubjectsFunction = mySampleSizeCalculationFunction, maxNumberOfIterations = 50)

## End(Not run)
```

`getSimulationSurvival` *Get Simulation Survival*

Description

Returns the analysis times, power, stopping probabilities, conditional power, and expected sample size for testing the hazard ratio in a two treatment groups survival design.

Usage

```
getSimulationSurvival(
  design = NULL,
  ...,
  thetaH0 = 1,
  directionUpper = NA,
  pi1 = NA_real_,
  pi2 = NA_real_,
  lambda1 = NA_real_,
  lambda2 = NA_real_,
  median1 = NA_real_,
  median2 = NA_real_,
  hazardRatio = NA_real_,
  kappa = 1,
  piecewiseSurvivalTime = NA_real_,
  allocation1 = 1,
  allocation2 = 1,
  eventTime = 12,
  accrualTime = c(0, 12),
  accrualIntensity = 0.1,
  accrualIntensityType = c("auto", "absolute", "relative"),
  dropoutRate1 = 0,
  dropoutRate2 = 0,
  dropoutTime = 12,
  maxNumberOfSubjects = NA_real_,
  plannedEvents = NA_real_,
  minNumberOfEventsPerStage = NA_real_,
  maxNumberOfEventsPerStage = NA_real_,
  conditionalPower = NA_real_,
  thetaH1 = NA_real_,
  maxNumberOfIterations = 1000L,
  maxNumberOfRawDatasetsPerStage = 0,
  longTimeSimulationAllowed = FALSE,
  seed = NA_real_,
```

```
  calcEventsFunction = NULL,
  showStatistics = FALSE
)
```

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
thetaH0	The null hypothesis value, default is 0 for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio). For non-inferiority designs, thetaH0 is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value != 0 (or a value != 1 for testing the mean ratio) can be specified. • *rates*: a value != 0 (or a value != 1 for testing the risk ratio pi1 / pi2) can be specified. • *survival data*: a bound for testing H0: hazard ratio = thetaH0 != 1 can be specified. • *count data*: a bound for testing H0: lambda1 / lambda2 = thetaH0 != 1 can be specified. For testing a rate in one sample, a value thetaH0 in (0, 1) has to be specified for defining the null hypothesis H0: pi = thetaH0.
directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.
pi1	A numeric value or vector that represents the assumed event rate in the treatment group, default is seq(0.2, 0.5, 0.1) (power calculations and simulations) or seq(0.4, 0.6, 0.1) (sample size calculations).
pi2	A numeric value that represents the assumed event rate in the control group, default is 0.2.
lambda1	The assumed hazard rate in the treatment group, there is no default. lambda1 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
lambda2	The assumed hazard rate in the reference group, there is no default. lambda2 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.
median1	The assumed median survival time in the treatment group, there is no default.
median2	The assumed median survival time in the reference group, there is no default. Must be a positive numeric of length 1.
hazardRatio	The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.

kappa
: A numeric value > 0. A kappa != 1 will be used for the specification of the shape of the Weibull distribution. Default is 1, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only piecewiselambda (as a single value) and kappa can be specified. This function is equivalent to pweibull(t, shape = kappa, scale = 1 / lambda) of the stats package, i.e., the scale parameter is 1 / 'hazard rate'.
For example, getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2) and pweibull(q = 130, shape = 4.2, scale = 1 / 0.01) provide the same result.

piecewiseSurvivalTime
: A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function
(for details see getPiecewiseSurvivalTime()).

allocation1
: The number how many subjects are assigned to treatment 1 in a subsequent order, default is 1

allocation2
: The number how many subjects are assigned to treatment 2 in a subsequent order, default is 1

eventTime
: The assumed time under which the event rates are calculated, default is 12.

accrualTime
: The assumed accrual time intervals for the study, default is c(0, 12) (for details see getAccrualTime()).

accrualIntensity
: A numeric vector of accrual intensities, default is the relative intensity 0.1 (for details see getAccrualTime()).

accrualIntensityType
: A character value specifying the accrual intensity input type. Must be one of "auto", "absolute", or "relative"; default is "auto", i.e., if all values are < 1 the type is "relative", otherwise it is "absolute".

dropoutRate1
: The assumed drop-out rate in the treatment group, default is 0.

dropoutRate2
: The assumed drop-out rate in the control group, default is 0.

dropoutTime
: The assumed time for drop-out rates in the control and the treatment group, default is 12.

maxNumberOfSubjects
: maxNumberOfSubjects > 0 needs to be specified. If accrual time and accrual intensity are specified, this will be calculated. Must be a positive integer of length 1.

plannedEvents
: plannedEvents is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, it is the number of events for both treatment arms. For multi-arm designs, plannedEvents refers to the overall number of events for the selected arms plus control.

minNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector minNumberOfEventsPerStage with length kMax determines the minimum number of events per stage (i.e., not cumulated), the first element is not taken into account.

maxNumberOfEventsPerStage
: When performing a data driven sample size recalculation, the numeric vector maxNumberOfEventsPerStage with length kMax determines the maximum number of events per stage (i.e., not cumulated), the first element is not taken into account.

conditionalPower
: If conditionalPower together with minNumberOfSubjectsPerStage and maxNumberOfSubjectsPe (or minNumberOfEventsPerStage and maxNumberOfEventsPerStage for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify thetaH1 and stDevH1 (for simulating means), pi1H1 and pi2H1 (for simulating rates), or thetaH1 (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

thetaH1
: If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

maxNumberOfIterations
: The number of simulation iterations, default is 1000. Must be a positive integer of length 1.

maxNumberOfRawDatasetsPerStage
: The number of raw datasets per stage that shall be extracted and saved as data.frame, default is 0. getRawData() can be used to get the extracted raw data from the object.

longTimeSimulationAllowed
: Logical that indicates whether long time simulations that consumes more than 30 seconds are allowed or not, default is FALSE.

seed
: The seed to reproduce the simulation, default is a random seed.

calcEventsFunction
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, event number recalculation is performed with conditional power and specified minNumberOfEventsPerStage and maxNumberOfEventsPerStage (see details and examples).

showStatistics
: Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

Details

At given design the function simulates the power, stopping probabilities, conditional power, and expected sample size at given number of events, number of subjects, and parameter configuration. It also simulates the time when the required events are expected under the given assumptions (exponentially, piecewise exponentially, or Weibull distributed survival times and constant or non-constant piecewise accrual). Additionally, integers allocation1 and allocation2 can be specified that determine the number allocated to treatment group 1 and treatment group 2, respectively. More precisely, unequal randomization ratios must be specified via the two integer arguments allocation1 and allocation2 which describe how many subjects are consecutively enrolled in each group, respectively, before a subject is assigned to the other group. For example, the arguments allocation1 = 2, allocation2 = 1, maxNumberOfSubjects = 300 specify 2:1 randomization with 200 subjects randomized to intervention and 100 to control. (Caveat: Do not use allocation1 = 200, allocation2 = 100, maxNumberOfSubjects = 300 as this would imply that the 200 intervention subjects are enrolled prior to enrollment of any control subjects.)

conditionalPower
The definition of thetaH1 makes only sense if kMax > 1 and if conditionalPower, minNumberOfEventsPerStage, and maxNumberOfEventsPerStage are defined.

Note that numberOfSubjects, numberOfSubjects1, and numberOfSubjects2 in the output are the expected number of subjects.

calcEventsFunction
This function returns the number of events at given conditional power and conditional critical value for specified testing situation. The function might depend on variables stage, conditionalPower, thetaH0, plannedEvents, singleEventsPerStage, minNumberOfEventsPerStage, maxNumberOfEventsPerStage, allocationRatioPlanned, conditionalCriticalValue, The function has to contain the three-dots argument '...' (see examples).

Value

Returns a SimulationResults object. The following generics (R generic functions) are available for this object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

Piecewise survival time

The first element of the vector piecewiseSurvivalTime must be equal to 0. piecewiseSurvivalTime can also be a list that combines the definition of the time intervals and hazard rates in the reference group. The definition of the survival time in the treatment group is obtained by the specification of the hazard ratio (see examples for details).

Staggered patient entry

accrualTime is the time period of subjects' accrual in a study. It can be a value that defines the end of accrual or a vector. In this case, accrualTime can be used to define a non-constant accrual over time. For this, accrualTime is a vector that defines the accrual intervals. The first element of accrualTime must be equal to 0 and, additionally, accrualIntensity needs to be specified. accrualIntensity itself is a value or a vector (depending on the length of accrualTime) that defines the intensity how subjects enter the trial in the intervals defined through accrualTime.

accrualTime can also be a list that combines the definition of the accrual time and accrual intensity (see below and examples for details).

If the length of accrualTime and the length of accrualIntensity are the same (i.e., the end of accrual is undefined), maxNumberOfSubjects > 0 needs to be specified and the end of accrual is calculated. In that case, accrualIntensity is the number of subjects per time unit, i.e., the absolute accrual intensity.

If the length of accrualTime equals the length of accrualIntensity - 1 (i.e., the end of accrual is defined), maxNumberOfSubjects is calculated if the absolute accrual intensity is given. If all elements in accrualIntensity are smaller than 1, accrualIntensity defines the *relative* intensity how subjects enter the trial. For example, accrualIntensity = c(0.1, 0.2) specifies that in the second accrual interval the intensity is doubled as compared to the first accrual interval. The actual (absolute) accrual intensity is calculated for the calculated or given maxNumberOfSubjects. Note that the default is accrualIntensity = 0.1 meaning that the *absolute* accrual intensity will be calculated.

Simulation Data

The summary statistics "Simulated data" contains the following parameters: median range; mean +/-sd

`$show(showStatistics = FALSE)` or `$setShowStatistics(FALSE)` can be used to disable the output of the aggregated simulated data.

Example 1:

```
simulationResults <- getSimulationSurvival(maxNumberOfSubjects = 100, plannedEvents = 30)
simulationResults$show(showStatistics = FALSE)
```

Example 2:

```
simulationResults <- getSimulationSurvival(maxNumberOfSubjects = 100, plannedEvents = 30)
simulationResults$setShowStatistics(FALSE)
simulationResults
```

`getData()` can be used to get the aggregated simulated data from the object as `data.frame`. The data frame contains the following columns:

1. `iterationNumber`: The number of the simulation iteration.
2. `stageNumber`: The stage.
3. `pi1`: The assumed or derived event rate in the treatment group.
4. `pi2`: The assumed or derived event rate in the control group.
5. `hazardRatio`: The hazard ratio under consideration (if available).
6. `analysisTime`: The analysis time.
7. `numberOfSubjects`: The number of subjects under consideration when the (interim) analysis takes place.
8. `eventsPerStage1`: The observed number of events per stage in treatment group 1.
9. `eventsPerStage2`: The observed number of events per stage in treatment group 2.
10. `singleEventsPerStage`: The observed number of events per stage in both treatment groups.
11. `rejectPerStage`: 1 if null hypothesis can be rejected, 0 otherwise.
12. `futilityPerStage`: 1 if study should be stopped for futility, 0 otherwise.
13. `eventsNotAchieved`: 1 if number of events could not be reached with observed number of subjects, 0 otherwise.
14. `testStatistic`: The test statistic that is used for the test decision, depends on which design was chosen (group sequential, inverse normal, or Fisher combination test)'
15. `logRankStatistic`: Z-score statistic which corresponds to a one-sided log-rank test at considered stage.
16. `hazardRatioEstimateLR`: The estimated hazard ratio, derived from the log-rank statistic.
17. `trialStop`: `TRUE` if study should be stopped for efficacy or futility or final stage, `FALSE` otherwise.
18. `conditionalPowerAchieved`: The conditional power for the subsequent stage of the trial for selected sample size and effect. The effect is either estimated from the data or can be user defined with `thetaH1`.

Raw Data

getRawData() can be used to get the simulated raw data from the object as data.frame. Note that getSimulationSurvival() must called before with maxNumberOfRawDatasetsPerStage > 0.

What maxNumberOfRawDatasetsPerStage **does**

When maxNumberOfRawDatasetsPerStage = 0 (the default), simulations run as usual but *no* patient-level ("raw") data are kept - only summary results.

If you set maxNumberOfRawDatasetsPerStage > 0, rpact will **save up to that many full patient-level datasets *per stage*** (i.e., per interim/final look). Each saved dataset corresponds to one simulated iteration and contains all subject-wise records accrued up to the stage at which that iteration stopped. You can later retrieve these datasets with getRawData().

Why "max" and not numberOfRawDatasetsPerStage**?**

The value is an *upper bound per stage*, not a fixed count. The actual number of datasets stored can be smaller because:

- Some iterations stop early (e.g., at stage 1), so later stages receive fewer datasets.
- The simulation might finish before reaching the cap due to other stopping or iteration limits.

As a result, the number specified is the **maximum possible** datasets saved *per stage*, not the exact number.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

Examples

```
## Not run:
# Fixed sample size with minimum required definitions, pi1 = (0.3,0.4,0.5,0.6) and
# pi2 = 0.3 at event time 12, and accrual time 24
getSimulationSurvival(
    pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, eventTime = 12,
    accrualTime = 24, plannedEvents = 40, maxNumberOfSubjects = 200,
    maxNumberOfIterations = 10
)

# Increase number of simulation iterations
getSimulationSurvival(
    pi1 = seq(0.3, 0.6, 0.1), pi2 = 0.3, eventTime = 12,
    accrualTime = 24, plannedEvents = 40, maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)

# Determine necessary accrual time with default settings if 200 subjects and
# 30 subjects per time unit can be recruited
getSimulationSurvival(
    plannedEvents = 40, accrualTime = 0,
    accrualIntensity = 30, maxNumberOfSubjects = 200, maxNumberOfIterations = 50
)
```

```
# Determine necessary accrual time with default settings if 200 subjects and
# if the first 6 time units 20 subjects per time unit can be recruited,
# then 30 subjects per time unit
getSimulationSurvival(
    plannedEvents = 40, accrualTime = c(0, 6),
    accrualIntensity = c(20, 30), maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)

# Determine maximum number of Subjects with default settings if the first
# 6 time units 20 subjects per time unit can be recruited, and after
# 10 time units 30 subjects per time unit
getSimulationSurvival(
    plannedEvents = 40, accrualTime = c(0, 6, 10),
    accrualIntensity = c(20, 30), maxNumberOfIterations = 50
)

# Specify accrual time as a list
at <- list(
    "0 - <6"  = 20,
    "6 - Inf" = 30
)
getSimulationSurvival(
    plannedEvents = 40, accrualTime = at,
    maxNumberOfSubjects = 200, maxNumberOfIterations = 50
)

# Specify accrual time as a list, if maximum number of subjects need to be calculated
at <- list(
    "0 - <6"   = 20,
    "6 - <=10" = 30
)
getSimulationSurvival(plannedEvents = 40, accrualTime = at, maxNumberOfIterations = 50)

# Specify effect size for a two-stage group sequential design with
# O'Brien & Fleming boundaries. Effect size is based on event rates
# at specified event time, directionUpper = FALSE needs to be specified
# because it should be shown that hazard ratio < 1
designGS <- getDesignGroupSequential(kMax = 2)
getSimulationSurvival(
    design = designGS,
    pi1 = 0.2, pi2 = 0.3, eventTime = 24, plannedEvents = c(20, 40),
    maxNumberOfSubjects = 200, directionUpper = FALSE, maxNumberOfIterations = 50
)

# As above, but with a three-stage O'Brien and Fleming design with
# specified information rates, note that planned events consists of integer values
designGS2 <- getDesignGroupSequential(informationRates = c(0.4, 0.7, 1))
getSimulationSurvival(
    design = designGS2,
    pi1 = 0.2, pi2 = 0.3, eventTime = 24,
    plannedEvents = round(designGS2$informationRates * 40),
    maxNumberOfSubjects = 200, directionUpper = FALSE,
    maxNumberOfIterations = 50
)

# Effect size is based on event rate at specified event time for the reference
```

```
# group and hazard ratio, directionUpper = FALSE needs to be specified because
# it should be shown that hazard ratio < 1
getSimulationSurvival(
    design = designGS, hazardRatio = 0.5,
    pi2 = 0.3, eventTime = 24, plannedEvents = c(20, 40), maxNumberOfSubjects = 200,
    directionUpper = FALSE, maxNumberOfIterations = 50
)

# Effect size is based on hazard rate for the reference group and
# hazard ratio, directionUpper = FALSE needs to be specified because
# it should be shown that hazard ratio < 1
getSimulationSurvival(
    design = designGS,
    hazardRatio = 0.5, lambda2 = 0.02, plannedEvents = c(20, 40),
    maxNumberOfSubjects = 200, directionUpper = FALSE,
    maxNumberOfIterations = 50
)

# Specification of piecewise exponential survival time and hazard ratios,
# note that in getSimulationSurvival only on hazard ratio is used
# in the case that the survival time is piecewise expoential
getSimulationSurvival(
    design = designGS,
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    hazardRatio = 1.5, plannedEvents = c(20, 40), maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)

pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04
)
getSimulationSurvival(
    design = designGS,
    piecewiseSurvivalTime = pws, hazardRatio = c(1.5),
    plannedEvents = c(20, 40), maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)

# Specification of piecewise exponential survival time for both treatment arms
getSimulationSurvival(
    design = designGS,
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    lambda1 = c(0.015, 0.03, 0.06), plannedEvents = c(20, 40),
    maxNumberOfSubjects = 200, maxNumberOfIterations = 50
)

# Specification of piecewise exponential survival time as a list,
# note that in getSimulationSurvival only on hazard ratio
# (not a vector) can be used
pws <- list(
    "0 - <5"  = 0.01,
    "5 - <10" = 0.02,
    ">=10"    = 0.04
)
getSimulationSurvival(
```

```
    design = designGS,
    piecewiseSurvivalTime = pws, hazardRatio = 1.5,
    plannedEvents = c(20, 40), maxNumberOfSubjects = 200,
    maxNumberOfIterations = 50
)

# Specification of piecewise exponential survival time and delayed effect
# (response after 5 time units)
getSimulationSurvival(
    design = designGS,
    piecewiseSurvivalTime = c(0, 5, 10), lambda2 = c(0.01, 0.02, 0.04),
    lambda1 = c(0.01, 0.02, 0.06), plannedEvents = c(20, 40),
    maxNumberOfSubjects = 200, maxNumberOfIterations = 50
)

# Specify effect size based on median survival times
getSimulationSurvival(
    median1 = 5, median2 = 3, plannedEvents = 40,
    maxNumberOfSubjects = 200, directionUpper = FALSE,
    maxNumberOfIterations = 50
)

# Specify effect size based on median survival
# times of Weibull distribtion with kappa = 2
getSimulationSurvival(
    median1 = 5, median2 = 3, kappa = 2,
    plannedEvents = 40, maxNumberOfSubjects = 200,
    directionUpper = FALSE, maxNumberOfIterations = 50
)

# Perform recalculation of number of events based on conditional power for a
# three-stage design with inverse normal combination test, where the conditional power
# is calculated under the specified effect size thetaH1 = 1.3 and up to a four-fold
# increase in originally planned sample size (number of events) is allowed.
# Note that the first value in minNumberOfEventsPerStage and
# maxNumberOfEventsPerStage is arbitrary, i.e., it has no effect.
designIN <- getDesignInverseNormal(informationRates = c(0.4, 0.7, 1))

resultsWithSSR1 <- getSimulationSurvival(
    design = designIN,
    hazardRatio = seq(1, 1.6, 0.1),
    pi2 = 0.3, conditionalPower = 0.8, thetaH1 = 1.3,
    plannedEvents = c(58, 102, 146),
    minNumberOfEventsPerStage = c(NA, 44, 44),
    maxNumberOfEventsPerStage = 4 * c(NA, 44, 44),
    maxNumberOfSubjects = 800, maxNumberOfIterations = 50
)
resultsWithSSR1

# If thetaH1 is unspecified, the observed hazard ratio estimate
# (calculated from the log-rank statistic) is used for performing the
# recalculation of the number of events
resultsWithSSR2 <- getSimulationSurvival(
    design = designIN,
    hazardRatio = seq(1, 1.6, 0.1),
    pi2 = 0.3, conditionalPower = 0.8, plannedEvents = c(58, 102, 146),
    minNumberOfEventsPerStage = c(NA, 44, 44),
```

```
    maxNumberOfEventsPerStage = 4 * c(NA, 44, 44),
    maxNumberOfSubjects = 800, maxNumberOfIterations = 50
)
resultsWithSSR2

# Compare it with design without event size recalculation
resultsWithoutSSR <- getSimulationSurvival(
    design = designIN,
    hazardRatio = seq(1, 1.6, 0.1), pi2 = 0.3,
    plannedEvents = c(58, 102, 145), maxNumberOfSubjects = 800,
    maxNumberOfIterations = 50
)
resultsWithoutSSR$overallReject
resultsWithSSR1$overallReject
resultsWithSSR2$overallReject

# Confirm that event size racalcuation increases the Type I error rate,
# i.e., you have to use the combination test
resultsWithSSRGS <- getSimulationSurvival(
    design = designGS2,
    hazardRatio = seq(1),
    pi2 = 0.3, conditionalPower = 0.8, plannedEvents = c(58, 102, 145),
    minNumberOfEventsPerStage = c(NA, 44, 44),
    maxNumberOfEventsPerStage = 4 * c(NA, 44, 44),
    maxNumberOfSubjects = 800, maxNumberOfIterations = 50
)
resultsWithSSRGS$overallReject

# Set seed to get reproducable results
identical(
    getSimulationSurvival(
        plannedEvents = 40, maxNumberOfSubjects = 200,
        seed = 99
    )$analysisTime,
    getSimulationSurvival(
        plannedEvents = 40, maxNumberOfSubjects = 200,
        seed = 99
    )$analysisTime
)

# Perform recalculation of number of events based on conditional power as above.
# The number of events is recalculated only in the first interim, the recalculated number
# is also used for the final stage. Here, we use the user defind calcEventsFunction as
# follows (note that the last stage value in minNumberOfEventsPerStage and maxNumberOfEventsPerStage
# has no effect):
myCalcEventsFunction <- function(...,
        stage, conditionalPower, estimatedTheta,
        plannedEvents, eventsOverStages,
        minNumberOfEventsPerStage, maxNumberOfEventsPerStage,
        conditionalCriticalValue) {
    theta <- max(1 + 1e-12, estimatedTheta)
    if (stage == 2) {
        requiredStageEvents <-
          max(0, conditionalCriticalValue + qnorm(conditionalPower))^2 * 4 / log(theta)^2
        requiredOverallStageEvents <- min(
            max(minNumberOfEventsPerStage[stage], requiredStageEvents),
            maxNumberOfEventsPerStage[stage]
```

```
        ) + eventsOverStages[stage - 1]
    } else {
      requiredOverallStageEvents <- 2 * eventsOverStages[stage - 1] - eventsOverStages[1]
    }
    return(requiredOverallStageEvents)
}
resultsWithSSR <- getSimulationSurvival(
    design = designIN,
    hazardRatio = seq(1, 2.6, 0.5),
    pi2 = 0.3,
    conditionalPower = 0.8,
    plannedEvents = c(58, 102, 146),
    minNumberOfEventsPerStage = c(NA, 44, 4),
    maxNumberOfEventsPerStage = 4 * c(NA, 44, 4),
    maxNumberOfSubjects = 800,
    calcEventsFunction = myCalcEventsFunction,
    seed = 1234,
    maxNumberOfIterations = 50
)

## End(Not run)
```

`getStageResults` *Get Stage Results*

Description

Returns summary statistics and p-values for a given data set and a given design.

Usage

```
getStageResults(
  design,
  dataInput,
  ...,
  stage = NA_integer_,
  directionUpper = NA
)
```

Arguments

`design`	The trial design.
`dataInput`	The summary data used for calculating the test results. This is either an element of `DatasetMeans`, of `DatasetRates`, or of `DatasetSurvival` and should be created with the function `getDataset()`. For more information see `getDataset()`.
`...`	Further (optional) arguments to be passed: `thetaH0` The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is 1 for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of

- *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified.
- *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified.
- *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified.

For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

`normalApproximation` The type of computation of the p-values. Default is `FALSE` for testing means (i.e., the t test is used) and `TRUE` for testing rates and the hazard ratio. For testing rates, if `normalApproximation = FALSE` is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting, `normalApproximation = FALSE` has no effect.

`equalVariances` The type of t test. For testing means in two treatment groups, either the t test assuming that the variances are equal or the t test without assuming this, i.e., the test of Welch-Satterthwaite is calculated, default is `TRUE`.

`intersectionTest` Defines the multiple test for the intersection hypotheses in the closed system of hypotheses when testing multiple hypotheses. Five options are available in multi-arm designs: `"Dunnett"`, `"Bonferroni"`, `"Simes"`, `"Sidak"`, and `"Hierarchical"`, default is `"Dunnett"`. Four options are available in population enrichment designs: `"SpiessensDebois"` (one subset only), `"Bonferroni"`, `"Simes"`, and `"Sidak"`, default is `"Simes"`.

`varianceOption` Defines the way to calculate the variance in multiple treatment arms (> 2) or population enrichment designs for testing means. For multiple arms, three options are available: `"overallPooled"`, `"pairwisePooled"`, and `"notPooled"`, default is `"overallPooled"`. For enrichment designs, the options are: `"pooled"`, `"pooledFromFull"` (one subset only), and `"notPooled"`, default is `"pooled"`.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. For testing means and rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is `TRUE`.

`stage` The stage number (optional). Default: total number of existing stages in the data input.

`directionUpper` Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.

Details

Calculates and returns the stage results of the specified design and data input at the specified stage.

Value

Returns a `StageResults` object.

- `names` to obtain the field names,
- `print()` to print the object,
- `summary()` to display a summary of the object,

- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

See Also

Other analysis functions: getAnalysisResults(), getClosedCombinationTestResults(), getClosedConditionalD
getConditionalPower(), getConditionalRejectionProbabilities(), getFinalConfidenceInterval(),
getFinalPValue(), getRepeatedConfidenceIntervals(), getRepeatedPValues(), getTestActions()

Examples

```
## Not run:
design <- getDesignInverseNormal()
dataRates <- getDataset(
    n1      = c(10, 10),
    n2      = c(20, 20),
    events1 = c( 8, 10),
    events2 = c(10, 16))
getStageResults(design, dataRates)

## End(Not run)
```

---

getSystemIdentifier *Get System Identifier*

Description

This function generates a unique system identifier based on the platform, R version, and rpact package version.

Usage

```
getSystemIdentifier(date = NULL)
```

Arguments

date	A character string or Date representing the date. Default is Sys.Date().

Value

A character string representing the unique system identifier.

Examples

```
## Not run:
getSystemIdentifier()

## End(Not run)
```

`getTestActions` *Get Test Actions*

Description

Returns test actions.

Usage

```
getTestActions(stageResults, ...)
```

Arguments

`stageResults`	The results at given stage, obtained from `getStageResults()`.
`...`	Only available for backward compatibility.

Details

Returns the test actions of the specified design and stage results at the specified stage.

Value

Returns a `character` vector of length `kMax` Returns a `numeric` vector of length `kMax` containing the test actions of each stage.

See Also

Other analysis functions: `getAnalysisResults()`, `getClosedCombinationTestResults()`, `getClosedConditionalD`
`getConditionalPower()`, `getConditionalRejectionProbabilities()`, `getFinalConfidenceInterval()`,
`getFinalPValue()`, `getRepeatedConfidenceIntervals()`, `getRepeatedPValues()`, `getStageResults()`

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
data <- getDataset(
    n      = c( 20,  30),
    means  = c( 50,  51),
    stDevs = c(130, 140)
)
getTestActions(getStageResults(design, dataInput = data))

## End(Not run)
```

getTestLabel *Get Test Label*

Description

Returns a string representation of the input value for use as a test label. Handles various types, including NULL, NA, vectors, and custom objects.

Usage

```
getTestLabel(x)
```

Arguments

x	The value to be converted into a test label.

Value

A character string representing the input value.

Examples

```
## Not run:
getTestLabel(NULL)
getTestLabel(NA)
getTestLabel(1:3)
getTestLabel(6)
getTestLabel(getDesignFisher())

## End(Not run)
```

getWideFormat *Get Wide Format*

Description

Returns the specified dataset as a `data.frame` in so-called wide format.

Usage

```
getWideFormat(dataInput)
```

Details

In the wide format (unstacked), the data are presented with each different data variable in a separate column, i.e., the different groups are in separate columns.

Value

A `data.frame` will be returned.

See Also

`getLongFormat()` for returning the dataset as a `data.frame` in long format.

`InstallationQualificationResult`
Installation Qualification Result Object

Description

This object represents the structured result of a full or partial installation qualification test execution. It includes metadata about the executed test suite, paths used, summary statistics, and status messages.

Format

An S3 object of class `InstallationQualificationResult` with the following elements:

completeUnitTestSetEnabled Logical indicating whether the full test set was enabled

testFileDirectory Directory containing test scripts

testFileTargetDirectory Directory to which tests are copied or linked

reportType Report type selected (`"compact"`, `"detailed"`, or `"Rout"`)

executionMode Execution mode (`"default"`, `"downloadOnly"`, `"downloadAndRunTests"`, or `"runTestsInTestFile`

scope Scope of the qualification (`"basic"`, `"devel"`, `"both"`, `"internet"`, or `"all"`)

resultDir Directory where the result reports are stored

resultOuputFile Main output report filename

reportFileNames Vector of report files generated

minNumberOfExpectedTests Minimum number of expected tests

totalNumberOfTests Number of tests actually run

numberOfFailedTests Number of failed tests

numberOfSkippedTests Number of skipped tests

resultMessage Message summarizing the result

statusMessage Detailed status message

status Overall result status (`"success"`, `"incomplete"`, or `"failed"`)

Details

The object is returned by the function `testPackage` and is of class `InstallationQualificationResult`.

See Also

`testPackage`

kableParameterSet *Create output in Markdown*

Description

The kable() function returns the output of the specified object formatted in Markdown.

Usage

```
## S3 method for class 'ParameterSet'
kable(x, ...)

## S3 method for class 'FieldSet'
kable(x, ..., enforceRowNames = TRUE, niceColumnNamesEnabled = TRUE)

## S3 method for class 'data.frame'
kable(x, ...)

## S3 method for class 'table'
kable(x, ...)

## S3 method for class 'matrix'
kable(x, ...)

## S3 method for class 'array'
kable(x, ...)

## S3 method for class 'numeric'
kable(x, ...)

## S3 method for class 'character'
kable(x, ...)

## S3 method for class 'logical'
kable(x, ...)

kable(x, ...)
```

Arguments

x	A ParameterSet. If x does not inherit from class ParameterSet, knitr::kable(x) will be returned.
...	Other arguments (see kable).

Details

This function is deprecated and should no longer be used. Manual use of kable() for rpact result objects is no longer needed, as the formatting and display will be handled automatically by the rpact package. Please remove any manual kable() calls from your code to avoid redundancy and potential issues. The results will be displayed in a consistent format automatically.

knit_print.FieldSet *Print Field Set in Markdown Code Chunks*

Description

The function knit_print.FieldSet is the default printing function for rpact result objects in knitr. The chunk option render uses this function by default. To fall back to the normal printing behavior set the chunk option render = normal_print. For more information see knit_print.

Usage

```
## S3 method for class 'FieldSet'
knit_print(x, ...)
```

Arguments

x	A FieldSet.
...	Other arguments (see knit_print).

Details

Generic function to print a field set in Markdown.

Markdown options

Use options("rpact.print.heading.base.number" = NUMBER) (where NUMBER is an integer value >= -2) to specify the heading level.

NUMBER = 1 results in the heading prefix #, NUMBER = 2 results in ##, ...

The default is options("rpact.print.heading.base.number" = -2), i.e., the top headings will be written italic but are not explicit defined as header. options("rpact.print.heading.base.number" = -1) means that all headings will be written bold but are not explicit defined as header.

Furthermore the following options can be set globally:

- rpact.auto.markdown.all: if TRUE, all output types will be rendered in Markdown format automatically.
- rpact.auto.markdown.print: if TRUE, all print outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.summary: if TRUE, all summary outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.plot: if TRUE, all plot outputs will be rendered in Markdown format automatically.

Example: options("rpact.auto.markdown.plot" = FALSE) disables the automatic knitting of plots inside Markdown documents.

knit_print.ParameterSet

Print Parameter Set in Markdown Code Chunks

Description

The function knit_print.ParameterSet is the default printing function for rpact result objects in knitr. The chunk option render uses this function by default. To fall back to the normal printing behavior set the chunk option render = normal_print. For more information see knit_print.

Usage

```
## S3 method for class 'ParameterSet'
knit_print(x, ...)
```

Arguments

x	A ParameterSet.
...	Other arguments (see knit_print).

Details

Generic function to print a parameter set in Markdown.

Markdown options

Use options("rpact.print.heading.base.number" = NUMBER) (where NUMBER is an integer value >= -2) to specify the heading level.

NUMBER = 1 results in the heading prefix #, NUMBER = 2 results in ##, ...

The default is options("rpact.print.heading.base.number" = -2), i.e., the top headings will be written italic but are not explicit defined as header. options("rpact.print.heading.base.number" = -1) means that all headings will be written bold but are not explicit defined as header.

Furthermore the following options can be set globally:

- rpact.auto.markdown.all: if TRUE, all output types will be rendered in Markdown format automatically.
- rpact.auto.markdown.print: if TRUE, all print outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.summary: if TRUE, all summary outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.plot: if TRUE, all plot outputs will be rendered in Markdown format automatically.

Example: options("rpact.auto.markdown.plot" = FALSE) disables the automatic knitting of plots inside Markdown documents.

knit_print.SummaryFactory

Print Summary Factory in Markdown Code Chunks

Description

The function knit_print.SummaryFactory is the default printing function for rpact summary objects in knitr. The chunk option render uses this function by default. To fall back to the normal printing behavior set the chunk option render = normal_print. For more information see knit_print.

Usage

```
## S3 method for class 'SummaryFactory'
knit_print(x, ...)
```

Arguments

x	A SummaryFactory.
...	Other arguments (see knit_print).

Details

Generic function to print a summary object in Markdown.

Markdown options

Use options("rpact.print.heading.base.number" = NUMBER) (where NUMBER is an integer value >= -2) to specify the heading level.

NUMBER = 1 results in the heading prefix #, NUMBER = 2 results in ##, ...

The default is options("rpact.print.heading.base.number" = -2), i.e., the top headings will be written italic but are not explicit defined as header. options("rpact.print.heading.base.number" = -1) means that all headings will be written bold but are not explicit defined as header.

Furthermore the following options can be set globally:

- rpact.auto.markdown.all: if TRUE, all output types will be rendered in Markdown format automatically.
- rpact.auto.markdown.print: if TRUE, all print outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.summary: if TRUE, all summary outputs will be rendered in Markdown format automatically.
- rpact.auto.markdown.plot: if TRUE, all plot outputs will be rendered in Markdown format automatically.

Example: options("rpact.auto.markdown.plot" = FALSE) disables the automatic knitting of plots inside Markdown documents.

length.TrialDesignSet *Length of Trial Design Set*

Description

Returns the number of designs in a `TrialDesignSet`.

Usage

```
## S3 method for class 'TrialDesignSet'
length(x)
```

Arguments

`x`	A `TrialDesignSet` object.

Details

Is helpful for iteration over all designs in a design set.

Value

Returns a non-negative `integer` of length 1 representing the number of design in the `TrialDesignSet`.

Examples

```
## Not run:
designSet <- getDesignSet(design = getDesignGroupSequential(), alpha = c(0.01, 0.05))
length(designSet)

## End(Not run)
```

MarkdownReporter *Markdown Reporter for Test Results*

Description

This class defines a Markdown reporter for test results, inheriting from the `R6::Reporter` class. It logs test results in Markdown format and saves them to a file named `test_results.md`.

Fields

`startTime` The start time of the test run.

`output` A character vector to store the log output.

`failures` The number of test failures.

`fileName` The name of the current test file being processed.

Methods

`initialize(...)` Initializes the reporter, setting up the output and failures fields.

`log(...)` Logs messages to the output field.

`start_reporter()` Starts the reporter, logging the introduction and test results header.

`start_file(file)` Sets the current file name being processed.

`getContext()` Gets the context from the current file name.

`add_result(context, test, result)` Adds a test result to the log, marking it as passed or failed.

`end_reporter()` Ends the reporter, logging the summary and saving the output to a file.

`finalize()` Finalizes the reporter, displaying a message that the test results were saved.

---

mvnprd *Original Algorithm AS 251: Normal Distribution*

Description

Calculates the Multivariate Normal Distribution with Product Correlation Structure published by Charles Dunnett, Algorithm AS 251.1 Appl.Statist. (1989), Vol.38, No.3, doi:10.2307/2347754.

Usage

```
mvnprd(..., A, B, BPD, EPS = 1e-06, INF, IERC = 1, HINC = 0)
```

Arguments

`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`A`	Upper limits of integration. Array of N dimensions
`B`	Lower limits of integration. Array of N dimensions
`BPD`	Values defining correlation structure. Array of N dimensions
`EPS`	desired accuracy. Defaults to 1e-06
`INF`	Determines where integration is done to infinity. Array of N dimensions. Valid values for INF(I): 0 = c(B(I), Inf), 1 = c(-Inf, A(I)), 2 = c(B(I), A(I))
`IERC`	error control. If set to 1, strict error control based on fourth derivative is used. If set to zero, error control based on halving intervals is used
`HINC`	Interval width for Simpson's rule. Value of zero caused a default .24 to be used

Details

This is a wrapper function for the original Fortran 77 code. For a multivariate normal vector with correlation structure defined by RHO(I,J) = BPD(I) * BPD(J), computes the probability that the vector falls in a rectangle in n-space with error less than eps.

mvstud *Original Algorithm AS 251: Student T Distribution*

Description

Calculates the Multivariate Normal Distribution with Product Correlation Structure published by Charles Dunnett, Algorithm AS 251.1 Appl.Statist. (1989), Vol.38, No.3, doi:10.2307/2347754.

Usage

```
mvstud(..., NDF, A, B, BPD, D, EPS = 1e-06, INF, IERC = 1, HINC = 0)
```

Arguments

...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
NDF	Degrees of Freedom. Use 0 for infinite D.F.
A	Upper limits of integration. Array of N dimensions
B	Lower limits of integration. Array of N dimensions
BPD	Values defining correlation structure. Array of N dimensions
D	Non-Centrality Vector
EPS	desired accuracy. Defaults to 1e-06
INF	Determines where integration is done to infinity. Array of N dimensions. Valid values for INF(I): 0 = c(B(I), Inf), 1 = c(-Inf, A(I)), 2 = c(B(I), A(I))
IERC	error control. If set to 1, strict error control based on fourth derivative is used. If set to zero, error control based on halving intervals is used
HINC	Interval width for Simpson's rule. Value of zero caused a default .24 to be used

Details

This is a wrapper function for the original Fortran 77 code. For a multivariate normal vector with correlation structure defined by RHO(I,J) = BPD(I) * BPD(J), computes the probability that the vector falls in a rectangle in n-space with error less than eps.

Examples

```
## Not run:
N <- 3
RHO <- 0.5
B <- rep(-5.0, length = N)
A <- rep(5.0, length = N)
INF <- rep(2, length = N)
BPD <- rep(sqrt(RHO), length = N)
D <- rep(0.0, length = N)
result <- mvstud(NDF = 0, A = A, B = B, BPD = BPD, INF = INF, D = D)
result

## End(Not run)
```

names.AnalysisResults *Names of a Analysis Results Object*

Description

Function to get the names of an AnalysisResults object.

Usage

```
## S3 method for class 'AnalysisResults'
names(x)
```

Arguments

x	An AnalysisResults object created by getAnalysisResults().

Details

Returns the names of an analysis results that can be accessed by the user.

Value

Returns a character vector containing the names of the AnalysisResults object.

names.FieldSet *Names of a Field Set Object*

Description

Function to get the names of a FieldSet object.

Usage

```
## S3 method for class 'FieldSet'
names(x)
```

Arguments

x	A FieldSet object.

Details

Returns the names of a field set that can be accessed by the user.

Value

Returns a character vector containing the names of the AnalysisResults object.

names.SimulationResults *Names of a Simulation Results Object*

Description

Function to get the names of a SimulationResults object.

Usage

```
## S3 method for class 'SimulationResults'
names(x)
```

Arguments

x	A SimulationResults object created by getSimulationResults[MultiArm/Enrichment][Means/

Details

Returns the names of a simulation results that can be accessed by the user.

Value

Returns a character vector containing the names of the AnalysisResults object.

names.StageResults *Names of a Stage Results Object*

Description

Function to get the names of a StageResults object.

Usage

```
## S3 method for class 'StageResults'
names(x)
```

Arguments

x	A StageResults object.

Details

Returns the names of stage results that can be accessed by the user.

Value

Returns a character vector containing the names of the AnalysisResults object.

names.TrialDesignSet *Names of a Trial Design Set Object*

Description

Function to get the names of a TrialDesignSet object.

Usage

```
## S3 method for class 'TrialDesignSet'
names(x)
```

Arguments

x	A TrialDesignSet object.

Details

Returns the names of a design set that can be accessed by the user.

Value

Returns a character vector containing the names of the AnalysisResults object.

Examples

```
## Not run:
designSet <- getDesignSet(design = getDesignGroupSequential(), alpha = c(0.01, 0.05))
names(designSet)

## End(Not run)
```

NumberOfSubjects *Number Of Subjects*

Description

Class for the definition of number of subjects results.

Details

NumberOfSubjects is a class for the definition of number of subjects results.

Fields

`time` The time values. Is a numeric vector.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`numberOfSubjects` In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

`obtain` *Extract a single parameter*

Description

Fetch a parameter from a parameter set.

Usage

```
obtain(x, ..., output)

## S3 method for class 'ParameterSet'
obtain(x, ..., output = c("named", "labeled", "value", "list"))

fetch(x, ..., output)

## S3 method for class 'ParameterSet'
fetch(x, ..., output = c("named", "labeled", "value", "list"))
```

Arguments

`x`	The `ParameterSet` object to fetch from.
`...`	One or more variables specified as: • a literal variable name • a positive integer, giving the position counting from the left • a negative integer, giving the position counting from the right. The default returns the last parameter. This argument is taken by expression and supports quasi-quotation (you can unquote column names and column locations).
`output`	A character defining the output type as follows: • "named" (default) returns the named value if the value is a single value, the value inside a named list otherwise • "value" returns only the value itself • "list" returns the value inside a named list

Examples

```
## Not run:
getDesignInverseNormal() |> fetch(kMax)
getDesignInverseNormal() |> fetch(kMax, output = "list")

## End(Not run)
```

ParameterSet *Parameter Set*

Description

Basic class for parameter sets.

Details

The parameter set implements basic functions for a set of parameters.

---

param_accrualIntensity *Parameter Description: Accrual Intensity*

Description

Parameter Description: Accrual Intensity

Arguments

`accrualIntensity`
: A numeric vector of accrual intensities, default is the relative intensity `0.1` (for details see `getAccrualTime()`).

---

param_accrualIntensityType *Parameter Description: Accrual Intensity Type*

Description

Parameter Description: Accrual Intensity Type

Arguments

`accrualIntensityType`
: A character value specifying the accrual intensity input type. Must be one of `"auto"`, `"absolute"`, or `"relative"`; default is `"auto"`, i.e., if all values are < 1 the type is `"relative"`, otherwise it is `"absolute"`.

param_accrualIntensity_counts *Parameter Description: accrualIntensity for Counts*

Description

Parameter Description: accrualIntensity for Counts

Arguments

`accrualIntensity` If specified, the assumed accrual intensities for the study, there is no default.

param_accrualTime *Parameter Description: Accrual Time*

Description

Parameter Description: Accrual Time

Arguments

`accrualTime` The assumed accrual time intervals for the study, default is `c(0, 12)` (for details see `getAccrualTime()`).

param_accrualTime_counts *Parameter Description: accrualTime for Counts*

Description

Parameter Description: accrualTime for Counts

Arguments

`accrualTime` If specified, the assumed accrual time interval(s) for the study, there is no default.

param_activeArms *Parameter Description: Active Arms*

Description

Parameter Description: Active Arms

Arguments

`activeArms` The number of active treatment arms to be compared with control, default is `3`.

param_adaptations *Parameter Description: Adaptations*

Description

Parameter Description: Adaptations

Arguments

adaptations	A logical vector of length kMax - 1 indicating whether or not an adaptation takes place at interim k, default is rep(TRUE, kMax - 1).

param_allocationRatioPlanned *Parameter Description: Allocation Ratio Planned*

Description

Parameter Description: Allocation Ratio Planned

Arguments

allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

param_allocationRatioPlanned_sampleSize *Parameter Description: Allocation Ratio Planned With Optimum Option*

Description

Parameter Description: Allocation Ratio Planned With Optimum Option

Arguments

allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. If allocationRatioPlanned = 0 is entered, the optimal allocation ratio yielding the smallest overall sample size is determined.

param_alpha *Parameter Description: Alpha*

Description

Parameter Description: Alpha

Arguments

alpha	The significance level alpha, default is 0.025. Must be a positive numeric of length 1.

param_alternative *Parameter Description: Alternative*

Description

Parameter Description: Alternative

Arguments

alternative	The alternative hypothesis value for testing means. This can be a vector of assumed alternatives, default is seq(0, 1, 0.2) (power calculations) or seq(0.2, 1, 0.2) (sample size calculations).

param_alternative_simulation *Parameter Description: Alternative for Simulation*

Description

Parameter Description: Alternative for Simulation

Arguments

alternative	The alternative hypothesis value for testing means under which the data is simulated. This can be a vector of assumed alternatives, default is seq(0, 1, 0.2).

param_beta *Parameter Description: Beta*

Description

Parameter Description: Beta

Arguments

beta	Type II error rate, necessary for providing sample size calculations (e.g., getSampleSizeMeans()), beta spending function designs, or optimum designs, default is 0.20. Must be a positive numeric of length 1.

param_bindingFutility *Parameter Description: Binding Futility*

Description

Parameter Description: Binding Futility

Arguments

`bindingFutility`
: Logical. If `bindingFutility = TRUE` is specified the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`).

param_calcEventsFunction *Parameter Description: Calculate Events Function*

Description

Parameter Description: Calculate Events Function

Arguments

`calcEventsFunction`
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, event number recalculation is performed with conditional power and specified `minNumberOfEventsPerStage` and `maxNumberOfEventsPerStage` (see details and examples).

param_calcSubjectsFunction *Parameter Description: Calculate Subjects Function*

Description

Parameter Description: Calculate Subjects Function

Arguments

`calcSubjectsFunction`
: Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, sample size recalculation is performed with conditional power and specified `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerSt` (see details and examples).

`param_conditionalPower` *Parameter Description: Conditional Power*

Description

Parameter Description: Conditional Power

Arguments

`conditionalPower`
: The conditional power for the subsequent stage under which the sample size recalculation is performed. Must be a positive numeric of length 1.

`param_conditionalPowerSimulation` *Parameter Description: Conditional Power*

Description

Parameter Description: Conditional Power

Arguments

`conditionalPower`
: If `conditionalPower` together with `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPe` (or `minNumberOfEventsPerStage` and `maxNumberOfEventsPerStage` for survival designs) is specified, a sample size recalculation based on the specified conditional power is performed. It is defined as the power for the subsequent stage given the current data. By default, the conditional power will be calculated under the observed effect size. Optionally, you can also specify `thetaH1` and `stDevH1` (for simulating means), `pi1H1` and `pi2H1` (for simulating rates), or `thetaH1` (for simulating hazard ratios) as parameters under which it is calculated and the sample size recalculation is performed.

`param_dataInput` *Parameter Description: Data Input*

Description

Parameter Description: Data Input

Arguments

`dataInput`
: The summary data used for calculating the test results. This is either an element of `DatasetMeans`, of `DatasetRates`, or of `DatasetSurvival` and should be created with the function `getDataset()`. For more information see `getDataset()`.

param_design *Parameter Description: Design*

Description

Parameter Description: Design

Arguments

design	The trial design.

param_design_with_default *Parameter Description: Design with Default*

Description

Parameter Description: Design with Default

Arguments

design	The trial design. If no trial design is specified, a fixed sample size design is used. In this case, Type I error rate alpha, Type II error rate beta, twoSidedPower, and sided can be directly entered as argument where necessary.

param_digits *Parameter Description: Digits*

Description

Parameter Description: Digits

Arguments

digits	Defines how many digits are to be used for numeric values. Must be a positive integer of length 1.

param_directionUpper *Parameter Description: Direction Upper*

Description

Parameter Description: Direction Upper

Arguments

directionUpper	Logical. Specifies the direction of the alternative, only applicable for one-sided testing; default is TRUE which means that larger values of the test statistics yield smaller p-values.

param_doseLevels *Parameter Description: Dose Levels*

Description

Parameter Description: Dose Levels

Arguments

doseLevels	The dose levels for the dose response relationship. If not specified, these dose levels are 1,...,activeArms.

param_dropoutRate1 *Parameter Description: Dropout Rate (1)*

Description

Parameter Description: Dropout Rate (1)

Arguments

dropoutRate1	The assumed drop-out rate in the treatment group, default is 0.

param_dropoutRate2 *Parameter Description: Dropout Rate (2)*

Description

Parameter Description: Dropout Rate (2)

Arguments

dropoutRate2	The assumed drop-out rate in the control group, default is 0.

param_dropoutTime *Parameter Description: Dropout Time*

Description

Parameter Description: Dropout Time

Arguments

dropoutTime	The assumed time for drop-out rates in the control and the treatment group, default is 12.

param_effectList *Parameter Description: Effect List*

Description

Parameter Description: Effect List

Arguments

effectList	List of subsets, prevalences, and effect sizes with columns and number of rows reflecting the different situations to consider (see examples).

param_effectMatrix *Parameter Description: Effect Matrix*

Description

Parameter Description: Effect Matrix

Arguments

effectMatrix	Matrix of effect sizes with activeArms columns and number of rows reflecting the different situations to consider.

param_effectMeasure *Parameter Description: Effect Measure*

Description

Parameter Description: Effect Measure

Arguments

effectMeasure	Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate (difference for means and rates or ratio for survival) ("effectEstimate"), default is "effectEstimate".

param_epsilonValue *Parameter Description: Epsilon Value*

Description

Parameter Description: Epsilon Value

Arguments

epsilonValue	For typeOfSelection = "epsilon" (select treatment arm / population not worse than epsilon compared to the best), the parameter epsilonValue has to be specified. Must be a numeric of length 1.

param_eventTime *Parameter Description: Event Time*

Description

Parameter Description: Event Time

Arguments

eventTime	The assumed time under which the event rates are calculated, default is 12.

param_fixedExposureTime_counts *Parameter Description: fixedExposureTime for Counts*

Description

Parameter Description: fixedExposureTime for Counts

Arguments

fixedExposureTime	If specified, the fixed time of exposure per subject for count data, there is no

param_grid *Parameter Description: Grid (Output Specification Of Multiple Plots)*

Description

Parameter Description: Grid (Output Specification Of Multiple Plots)

Arguments

grid	An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

param_groups *Parameter Description: Number Of Treatment Groups*

Description

Parameter Description: Number Of Treatment Groups

Arguments

groups	The number of treatment groups (1 or 2), default is 2.

param_hazardRatio *Parameter Description: Hazard Ratio*

Description

Parameter Description: Hazard Ratio

Arguments

hazardRatio	The vector of hazard ratios under consideration. If the event or hazard rates in both treatment groups are defined, the hazard ratio needs not to be specified as it is calculated, there is no default. Must be a positive numeric of length 1.

`param_includeAllParameters`

Parameter Description: Include All Parameters

Description

Parameter Description: Include All Parameters

Arguments

`includeAllParameters`

Logical. If `TRUE`, all available parameters will be included in the data frame; a meaningful parameter selection otherwise, default is `FALSE`.

`param_informationEpsilon`

Parameter Description: Information Epsilon

Description

Parameter Description: Information Epsilon

Arguments

`informationEpsilon`

Positive integer value specifying the absolute information epsilon, which defines the maximum distance from the observed information to the maximum information that causes the final analysis. Updates at the final analysis in case the observed information at the final analysis is smaller ("under-running") than the planned maximum information `maxInformation`, default is 0. Alternatively, a floating-point number > 0 and < 1 can be specified to define a relative information epsilon.

`param_informationRates`

Parameter Description: Information Rates

Description

Parameter Description: Information Rates

Arguments

`informationRates`

The information rates t_1, ..., t_kMax (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. For the weighted inverse normal design, the weights are derived through w_1 = sqrt(t_1), and w_k = sqrt(t_k - t_(k-1)). For the weighted Fisher's combination test, the weights (scales) are w_k = sqrt((t_k - t_(k-1)) / t_1) (see the documentation).

param_intersectionTest_Enrichment

Parameter Description: Intersection Test

Description

Parameter Description: Intersection Test

Arguments

`intersectionTest`

Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Four options are available in enrichment designs: `"SpiessensDebois"`, `"Bonferroni"`, `"Simes"`, and `"Sidak"`, default is `"Simes"`.

param_intersectionTest_MultiArm

Parameter Description: Intersection Test

Description

Parameter Description: Intersection Test

Arguments

`intersectionTest`

Defines the multiple test for the intersection hypotheses in the closed system of hypotheses. Five options are available in multi-arm designs: `"Dunnett"`, `"Bonferroni"`, `"Simes"`, `"Sidak"`, and `"Hierarchical"`, default is `"Dunnett"`.

param_kappa

Parameter Description: Kappa

Description

Parameter Description: Kappa

Arguments

`kappa`

A numeric value > 0. A `kappa != 1` will be used for the specification of the shape of the Weibull distribution. Default is `1`, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only `piecewiselambda` (as a single value) and `kappa` can be specified. This function is equivalent to `pweibull(t, shape = kappa, scale = 1 / lambda)` of the `stats` package, i.e., the scale parameter is `1 / 'hazard rate'`.
For example, `getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2)` and `pweibull(q = 130, shape = 4.2, scale = 1 / 0.01)` provide the same result.

param_kMax *Parameter Description: Maximum Number of Stages*

Description

Parameter Description: Maximum Number of Stages

Arguments

kMax	The maximum number of stages K. Must be a positive integer of length 1 (default value is 3). The maximum selectable kMax is 20 for group sequential or inverse normal and 6 for Fisher combination test designs.

param_lambda1 *Parameter Description: Lambda (1)*

Description

Parameter Description: Lambda (1)

Arguments

lambda1	The assumed hazard rate in the treatment group, there is no default. lambda1 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

param_lambda1_counts *Parameter Description: lambda (1) for Counts*

Description

Parameter Description: lambda (1) for Counts

Arguments

lambda1	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the active treatment group, there is no default.

param_lambda2 *Parameter Description: Lambda (2)*

Description

Parameter Description: Lambda (2)

Arguments

lambda2	The assumed hazard rate in the reference group, there is no default. lambda2 can also be used to define piecewise exponentially distributed survival times (see details). Must be a positive numeric of length 1.

param_lambda2_counts *Parameter Description: lambda (2) for Counts*

Description

Parameter Description: lambda (2) for Counts

Arguments

lambda2	A numeric value that represents the assumed rate of a homogeneous Poisson process in the control group, there is no default.

param_lambda_counts *Parameter Description: lambda for Counts*

Description

Parameter Description: lambda for Counts

Arguments

lambda	A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the pooled treatment groups, there is no default.

param_legendPosition *Parameter Description: Legend Position On Plots*

Description

Parameter Description: Legend Position On Plots

Arguments

legendPosition	The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually: • -1: no legend will be shown • NA: the algorithm tries to find a suitable position • 0: legend position outside plot • 1: legend position left top • 2: legend position left center • 3: legend position left bottom • 4: legend position right top • 5: legend position right center • 6: legend position right bottom

param_maxInformation *Parameter Description: Maximum Information*

Description

Parameter Description: Maximum Information

Arguments

`maxInformation` Positive value specifying the maximum information.

param_maxNumberOfEventsPerStage *Parameter Description: Max Number Of Events Per Stage*

Description

Parameter Description: Max Number Of Events Per Stage

Arguments

`maxNumberOfEventsPerStage`
When performing a data driven sample size recalculation, the numeric vector `maxNumberOfEventsPerStage` with length `kMax` determines the maximum number of events per stage (i.e., not cumulated), the first element is not taken into account.

param_maxNumberOfIterations *Parameter Description: Maximum Number Of Iterations*

Description

Parameter Description: Maximum Number Of Iterations

Arguments

`maxNumberOfIterations`
The number of simulation iterations, default is `1000`. Must be a positive integer of length 1.

`param_maxNumberOfSubjects`

Parameter Description: Maximum Number Of Subjects

Description

Parameter Description: Maximum Number Of Subjects

Arguments

`maxNumberOfSubjects`

`maxNumberOfSubjects > 0` needs to be specified for power calculations or calculation of necessary follow-up (count data). For two treatment arms, it is the maximum number of subjects for both treatment arms.

`param_maxNumberOfSubjectsPerStage`

Parameter Description: Maximum Number Of Subjects Per Stage

Description

Parameter Description: Maximum Number Of Subjects Per Stage

Arguments

`maxNumberOfSubjectsPerStage`

When performing a data driven sample size recalculation, the numeric vector `maxNumberOfSubjectsPerStage` with length `kMax` determines the maximum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs `maxNumberOfSubjectsPerStage` refers to the maximum number of subjects per selected active arm.

`param_maxNumberOfSubjects_survival`

Parameter Description: Maximum Number Of Subjects For Survival Endpoint

---

Description

Parameter Description: Maximum Number Of Subjects For Survival Endpoint

Arguments

`maxNumberOfSubjects`

`maxNumberOfSubjects > 0` needs to be specified. If accrual time and accrual intensity are specified, this will be calculated. Must be a positive integer of length 1.

param_median1 *Parameter Description: Median (1)*

Description

Parameter Description: Median (1)

Arguments

median1	The assumed median survival time in the treatment group, there is no default.

param_median2 *Parameter Description: Median (2)*

Description

Parameter Description: Median (2)

Arguments

median2	The assumed median survival time in the reference group, there is no default. Must be a positive numeric of length 1.

param_minNumberOfEventsPerStage *Parameter Description: Min Number Of Events Per Stage*

Description

Parameter Description: Min Number Of Events Per Stage

Arguments

minNumberOfEventsPerStage	When performing a data driven sample size recalculation, the numeric vector minNumberOfEventsPerStage with length kMax determines the minimum number of events per stage (i.e., not cumulated), the first element is not taken into account.

`param_minNumberOfSubjectsPerStage` *Parameter Description: Minimum Number Of Subjects Per Stage*

Description

Parameter Description: Minimum Number Of Subjects Per Stage

Arguments

`minNumberOfSubjectsPerStage`
: When performing a data driven sample size recalculation, the numeric vector `minNumberOfSubjectsPerStage` with length `kMax` determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs `minNumberOfSubjectsPerStage` refers to the minimum number of subjects per selected active arm.

`param_niceColumnNamesEnabled` *Parameter Description: Nice Column Names Enabled*

Description

Parameter Description: Nice Column Names Enabled

Arguments

`niceColumnNamesEnabled`
: Logical. If `TRUE`, nice looking column names will be used; syntactic names (variable names) otherwise (see `make.names`).

`param_nMax` *Parameter Description: N_max*

Description

Parameter Description: N_max

Arguments

`nMax`
: The maximum sample size. Must be a positive integer of length 1.

param_normalApproximation

Parameter Description: Normal Approximation

Description

Parameter Description: Normal Approximation

Arguments

`normalApproximation`
: The type of computation of the p-values. Default is `FALSE` for testing means (i.e., the t test is used) and `TRUE` for testing rates and the hazard ratio. For testing rates, if `normalApproximation = FALSE` is specified, the binomial test (one sample) or the exact test of Fisher (two samples) is used for calculating the p-values. In the survival setting `normalApproximation = FALSE` has no effect.

param_nPlanned

Parameter Description: N Planned

Description

Parameter Description: N Planned

Arguments

`nPlanned`
: The additional (i.e., "new" and not cumulative) sample size planned for each of the subsequent stages. The argument must be a vector with length equal to the number of remaining stages and contain the combined sample size from both treatment groups if two groups are considered. For survival outcomes, it should contain the planned number of additional events. For multi-arm designs, it is the per-comparison (combined) sample size. For enrichment designs, it is the (combined) sample size for the considered sub-population.

param_overdispersion_counts

Parameter Description: overdispersion for Counts

Description

Parameter Description: overdispersion for Counts

Arguments

`overdispersion`
: A numeric value that represents the assumed overdispersion of the negative binomial distribution, default is `0`.

param_palette *Parameter Description: Palette*

Description

Parameter Description: Palette

Arguments

palette	The palette, default is "Set1".

param_pi1_rates *Parameter Description: Pi (1) for Rates*

Description

Parameter Description: Pi (1) for Rates

Arguments

pi1	A numeric value or vector that represents the assumed probability in the active treatment group if two treatment groups are considered, or the alternative probability for a one treatment group design, default is seq(0.2, 0.5, 0.1) (power calculations and simulations) or seq(0.4, 0.6, 0.1) (sample size calculations).

param_pi1_survival *Parameter Description: Pi (1) for Survival Data*

Description

Parameter Description: Pi (1) for Survival Data

Arguments

pi1	A numeric value or vector that represents the assumed event rate in the treatment group, default is seq(0.2, 0.5, 0.1) (power calculations and simulations) or seq(0.4, 0.6, 0.1) (sample size calculations).

param_pi2_rates *Parameter Description: Pi (2) for Rates*

Description

Parameter Description: Pi (2) for Rates

Arguments

pi2	A numeric value that represents the assumed probability in the reference group if two treatment groups are considered, default is 0.2.

param_pi2_survival *Parameter Description: Pi (2) for Survival Data*

Description

Parameter Description: Pi (2) for Survival Data

Arguments

pi2	A numeric value that represents the assumed event rate in the control group, default is 0.2.

param_piecewiseSurvivalTime *Parameter Description: Piecewise Survival Time*

Description

Parameter Description: Piecewise Survival Time

Arguments

piecewiseSurvivalTime	A vector that specifies the time intervals for the piecewise definition of the exponential survival time cumulative distribution function (for details see getPiecewiseSurvivalTime()).

param_plannedCalendarTime *Parameter Description: Planned Calendar Time*

Description

Parameter Description: Planned Calendar Time

Arguments

plannedCalendarTime	For simulating count data, the time points where an analysis is planned to be performed. Should be a vector of length kMax

param_plannedEvents *Parameter Description: Planned Events*

Description

Parameter Description: Planned Events

Arguments

plannedEvents	plannedEvents is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, it is the number of events for both treatment arms. For multi-arm designs, plannedEvents refers to the overall number of events for the selected arms plus control.

param_plannedSubjects *Parameter Description: Planned Subjects*

Description

Parameter Description: Planned Subjects

Arguments

plannedSubjects	plannedSubjects is a numeric vector of length kMax (the number of stages of the design) that determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, it is the number of subjects for both treatment arms. For multi-arm designs, plannedSubjects refers to the number of subjects per selected active arm.

param_plotPointsEnabled *Parameter Description: Plot Points Enabled*

Description

Parameter Description: Plot Points Enabled

Arguments

plotPointsEnabled	Logical. If TRUE, additional points will be plotted.

param_plotSettings *Parameter Description: Plot Settings*

Description

Parameter Description: Plot Settings

Arguments

plotSettings	An object of class PlotSettings created by getPlotSettings().

param_populations *Parameter Description: Populations*

Description

Parameter Description: Populations

Arguments

populations	The number of populations in a two-sample comparison, default is 3.

param_rValue *Parameter Description: R Value*

Description

Parameter Description: R Value

Arguments

rValue	For typeOfSelection = "rbest" (select the rValue best treatment arms / populations), the parameter rValue has to be specified.

param_seed *Parameter Description: Seed*

Description

Parameter Description: Seed

Arguments

seed	The seed to reproduce the simulation, default is a random seed.

param_selectArmsFunction *Parameter Description: Select Arms Function*

Description

Parameter Description: Select Arms Function

Arguments

`selectArmsFunction`
: Optionally, a function can be entered that defines the way of how treatment arms are selected. This function is allowed to depend on `effectVector` with length `activeArms`, `stage`, `conditionalPower`, `conditionalCriticalValue`, `plannedSubjects/planne` `allocationRatioPlanned`, `selectedArms`, `thetaH1` (for means and survival), `stDevH1` (for means), `overallEffects`, and for rates additionally: `piTreatmentsH1`, `piControlH1`, `overallRates`, and `overallRatesControl` (see examples).

param_selectPopulationsFunction *Parameter Description: Select Populations Function*

Description

Parameter Description: Select Populations Function

Arguments

`selectPopulationsFunction`
: Optionally, a function can be entered that defines the way of how populations are selected. This function is allowed to depend on `effectVector` with length `populations` `stage`, `conditionalPower`, `conditionalCriticalValue`, `plannedSubjects/plann` `allocationRatioPlanned`, `selectedPopulations`, `thetaH1` (for means and survival), `stDevH1` (for means), `overallEffects`, and for rates additionally: `piTreatmentsH1`, `piControlH1`, `overallRates`, and `overallRatesControl` (see examples).

param_showSource *Parameter Description: Show Source*

Description

Parameter Description: Show Source

Arguments

showSource	Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values: • "commands": returns a character vector with plot commands • "axes": returns a list with the axes definitions • "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs) • "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs) Note: no plot object will be returned if showSource is a character.

---

param_showStatistics *Parameter Description: Show Statistics*

Description

Parameter Description: Show Statistics

Arguments

showStatistics	Logical. If TRUE, summary statistics of the simulated data are displayed for the print command, otherwise the output is suppressed, default is FALSE.

---

param_sided *Parameter Description: Sided*

Description

Parameter Description: Sided

Arguments

sided	Is the alternative one-sided (1) or two-sided (2), default is 1. Must be a positive integer of length 1.

---

param_slope *Parameter Description: Slope*

Description

Parameter Description: Slope

Arguments

slope	If typeOfShape = "sigmoidEmax" is selected, slope can be entered to specify the slope of the sigmoid Emax model, default is 1.

param_stage *Parameter Description: Stage*

Description

Parameter Description: Stage

Arguments

stage	The stage number (optional). Default: total number of existing stages in the data input.

param_stageResults *Parameter Description: Stage Results*

Description

Parameter Description: Stage Results

Arguments

stageResults	The results at given stage, obtained from getStageResults().

param_stDev *Parameter Description: Standard Deviation*

Description

Parameter Description: Standard Deviation

Arguments

stDev	The standard deviation under which the sample size or power calculation is performed, default is 1. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If meanRatio = TRUE is specified, stDev defines the coefficient of variation sigma / mu2.

param_stDevH1 *Parameter Description: Standard Deviation Under Alternative*

Description

Parameter Description: Standard Deviation Under Alternative

Arguments

stDevH1	If specified, the value of the standard deviation under which the conditional power or sample size recalculation calculation is performed, default is the value of stDev.

param_stDevSimulation *Parameter Description: Standard Deviation for Simulation*

Description

Parameter Description: Standard Deviation for Simulation

Arguments

stDev	The standard deviation under which the data is simulated, default is 1. For two-armed trials, it is allowed to specify the standard deviations separately, i.e., as vector with two elements. If meanRatio = TRUE is specified, stDev defines the coefficient of variation sigma / mu2.

param_stratifiedAnalysis *Parameter Description: Stratified Analysis*

Description

Parameter Description: Stratified Analysis

Arguments

stratifiedAnalysis	Logical. For enrichment designs, typically a stratified analysis should be chosen. For testing rates, also a non-stratified analysis based on overall data can be performed. For survival data, only a stratified analysis is possible (see Brannath et al., 2009), default is TRUE.

param_successCriterion *Parameter Description: Success Criterion*

Description

Parameter Description: Success Criterion

Arguments

successCriterion	Defines when the study is stopped for efficacy at interim. Two options are available: "all" stops the trial if the efficacy criterion is fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim, default is "all".

param_theta *Parameter Description: Theta*

Description

Parameter Description: Theta

Arguments

theta	A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.

param_thetaH0 *Parameter Description: Theta H0*

Description

Parameter Description: Theta H0

Arguments

thetaH0	The null hypothesis value, default is `0` for the normal and the binary case (testing means and rates, respectively), it is `1` for the survival case (testing the hazard ratio). For non-inferiority designs, `thetaH0` is the non-inferiority bound. That is, in case of (one-sided) testing of • *means*: a value `!= 0` (or a value `!= 1` for testing the mean ratio) can be specified. • *rates*: a value `!= 0` (or a value `!= 1` for testing the risk ratio `pi1 / pi2`) can be specified. • *survival data*: a bound for testing H0: `hazard ratio = thetaH0 != 1` can be specified. • *count data*: a bound for testing H0: `lambda1 / lambda2 = thetaH0 != 1` can be specified. For testing a rate in one sample, a value `thetaH0` in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.

param_thetaH1 *Parameter Description: Effect Under Alternative*

Description

Parameter Description: Effect Under Alternative

Arguments

thetaH1	If specified, the value of the alternative under which the conditional power or sample size recalculation calculation is performed. Must be a numeric of length 1.

param_theta_counts *Parameter Description: theta for Counts*

Description

Parameter Description: theta for Counts

Arguments

theta	A numeric value or vector that represents the assumed mean ratios lambda1/lambda2 of a homogeneous Poisson process, there is no default.

param_three_dots *Parameter Description: "..."*

Description

Parameter Description: "..."

Arguments

...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.

param_three_dots_plot *Parameter Description: "..." (optional plot arguments)*

Description

Parameter Description: "..." (optional plot arguments)

Arguments

...	Optional plot arguments. At the moment `xlim` and `ylim` are implemented for changing x or y axis limits without dropping data observations.

param_threshold *Parameter Description: Threshold*

Description

Parameter Description: Threshold

Arguments

threshold	Selection criterion: treatment arm / population is selected only if `effectMeasure` exceeds `threshold`, default is `-Inf`. `threshold` can also be a vector of length `activeArms` referring to a separate threshold condition over the treatment arms.

param_tolerance *Parameter Description: Tolerance*

Description

Parameter Description: Tolerance

Arguments

tolerance	The numerical tolerance, default is 1e-06. Must be a positive numeric of length 1.

param_typeOfComputation *Parameter Description: Type Of Computation*

Description

Parameter Description: Type Of Computation

Arguments

typeOfComputation	Three options are available: "Schoenfeld", "Freedman", "HsiehFreedman", the default is "Schoenfeld". For details, see Hsieh (Statistics in Medicine, 1992). For non-inferiority testing (i.e., thetaH0 != 1), only Schoenfeld's formula can be used.

param_typeOfDesign *Parameter Description: Type of Design*

Description

Parameter Description: Type of Design

Arguments

typeOfDesign	The type of design. Type of design is one of the following: O'Brien & Fleming ("OF"), Pocock ("P"), Wang & Tsiatis Delta class ("WT"), Pampallona & Tsiatis ("PT"), Haybittle & Peto ("HP"), Optimum design within Wang & Tsiatis class ("WToptimum"), O'Brien & Fleming type alpha spending ("asOF"), Pocock type alpha spending ("asP"), Kim & DeMets alpha spending ("asKD"), Hwang, Shi & DeCani alpha spending ("asHSD"), user defined alpha spending ("asUser"), no early efficacy stop ("noEarlyEfficacy"), default is "OF".

param_typeOfSelection *Parameter Description: Type of Selection*

Description

Parameter Description: Type of Selection

Arguments

`typeOfSelection`

The way the treatment arms or populations are selected at interim. Five options are available: `"best"`, `"rbest"`, `"epsilon"`, `"all"`, and `"userDefined"`, default is `"best"`.
For `"rbest"` (select the `rValue` best treatment arms/populations), the parameter `rValue` has to be specified, for `"epsilon"` (select treatment arm/population not worse than epsilon compared to the best), the parameter `epsilonValue` has to be specified. If `"userDefined"` is selected, `"selectArmsFunction"` or `"selectPopulationsFunction"` has to be specified.

param_typeOfShapeMeans *Parameter Description: Type Of Shape*

Description

Parameter Description: Type Of Shape

Arguments

`typeOfShape` The shape of the dose-response relationship over the treatment groups. This can be either `"linear"`, `"sigmoidEmax"`, or `"userDefined"`, default is `"linear"`.
For `"linear"`, `muMaxVector` specifies the range of effect sizes for the treatment group with highest response. If `"sigmoidEmax"` is selected, `gED50` and `slope` has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For `"sigmoidEmax"`, `muMaxVector` specifies the range of effect sizes for the treatment group with response according to infinite dose. If `"userDefined"` is selected, `effectMatrix` has to be entered.

param_typeOfShapeRates

Parameter Description: Type Of Shape

Description

Parameter Description: Type Of Shape

Arguments

typeOfShape	The shape of the dose-response relationship over the treatment groups. This can be either "linear", "sigmoidEmax", or "userDefined", default is "linear". For "linear", piMaxVector specifies the range of effect sizes for the treatment group with highest response. If "sigmoidEmax" is selected, gED50 and slope has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For "sigmoidEmax", piMaxVector specifies the range of effect sizes for the treatment group with response according to infinite dose. If "userDefined" is selected, effectMatrix has to be entered.

param_typeOfShapeSurvival

Parameter Description: Type Of Shape

Description

Parameter Description: Type Of Shape

Arguments

typeOfShape	The shape of the dose-response relationship over the treatment groups. This can be either "linear", "sigmoidEmax", or "userDefined", default is "linear". For "linear", omegaMaxVector specifies the range of effect sizes for the treatment group with highest response. If "sigmoidEmax" is selected, gED50 and slope has to be entered to specify the ED50 and the slope of the sigmoid Emax model. For "sigmoidEmax", omegaMaxVector specifies the range of effect sizes for the treatment group with response according to infinite dose. If "userDefined" is selected, effectMatrix has to be entered.

param_userAlphaSpending *Parameter Description: User Alpha Spending*

Description

Parameter Description: User Alpha Spending

Arguments

`userAlphaSpending`
: The user defined alpha spending. Numeric vector of length `kMax` containing the cumulative alpha-spending (Type I error rate) up to each interim stage: `0 <= alpha_1 <= ... <= alpha_K <= alpha`.

param_varianceOption *Parameter Description: Variance Option*

Description

Parameter Description: Variance Option

Arguments

`varianceOption`
: Defines the way to calculate the variance in multiple treatment arms (> 2) or population enrichment designs for testing means. For multiple arms, three options are available: `"overallPooled"`, `"pairwisePooled"`, and `"notPooled"`, default is `"overallPooled"`. For enrichment designs, the options are: `"pooled"`, `"pooledFromFull"` (one subset only), and `"notPooled"`, default is `"pooled"`.

PerformanceScore *Performance Score*

Description

Contains the conditional performance score, its sub-scores and components according to Herrmann et al. (2020) for a given simulation result.

Details

Use getPerformanceScore to calculate the performance score.

PiecewiseSurvivalTime *Piecewise Exponential Survival Time*

Description

Class for the definition of piecewise survival times.

Details

PiecewiseSurvivalTime is a class for the definition of piecewise survival times.

Fields

piecewiseSurvivalTime The time intervals for the piecewise definition of the exponential survival time cumulative distribution function. Is a numeric vector.

lambda1 The assumed hazard rate in the treatment group. Is a numeric vector of length kMax.

lambda2 The assumed hazard rate in the reference group. Is a numeric vector of length 1.

hazardRatio The hazard ratios under consideration. Is a numeric vector of length kMax.

pi1 The assumed event rate in the treatment group. Is a numeric vector of length kMax containing values between 0 and 1.

pi2 The assumed event rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

median1 The assumed median survival time in the treatment group. Is a numeric vector.

median2 The assumed median survival time in the reference group. Is a numeric vector of length 1.

eventTime The assumed time under which the event rates are calculated. Is a numeric vector of length 1.

kappa The shape of the Weibull distribution if kappa!=1. Is a numeric vector of length 1.

piecewiseSurvivalEnabled Indicates whether specification of piecewise definition of survival time is selected. Is a logical vector of length 1.

delayedResponseAllowed If TRUE, delayed response is allowed, if FALSE the response is not delayed.

delayedResponseEnabled If TRUE, delayed response is enabled, if FALSE delayed response is not enabled.

plot.AnalysisResults *Analysis Results Plotting*

Description

Plots the conditional power together with the likelihood function.

Usage

```
## S3 method for class 'AnalysisResults'
plot(
  x,
  y,
  ...,
  type = 1L,
  nPlanned = NA_real_,
  allocationRatioPlanned = NA_real_,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  legendTitle = NA_character_,
  palette = "Set1",
  legendPosition = NA_integer_,
  showSource = FALSE,
  grid = 1,
  plotSettings = NULL
)
```

Arguments

x	The analysis results at given stage, obtained from `getAnalysisResults()`.
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. Furthermore the following arguments can be defined: • `thetaRange`: A range of assumed effect sizes if testing means or a survival design was specified. Additionally, if testing means was selected, `assumedStDev` (assumed standard deviation) can be specified (default is `1`). • `piTreatmentRange`: A range of assumed rates pi1 to calculate the conditional power. Additionally, if a two-sample comparison was selected, `pi2` can be specified (default is the value from `getAnalysisResults()`). • `directionUpper`: Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values. • `thetaH0`: The null hypothesis value, default is `0` for the normal and the binary case, it is `1` for the survival case. For testing a rate in one sample, a value thetaH0 in (0, 1) has to be specified for defining the null hypothesis H0: `pi = thetaH0`.
type	The plot type (default = 1). Note that at the moment only one type (the conditional power plot) is available.
nPlanned	The additional (i.e., "new" and not cumulative) sample size planned for each of the subsequent stages. The argument must be a vector with length equal to the number of remaining stages and contain the combined sample size from both treatment groups if two groups are considered. For survival outcomes, it should contain the planned number of additional events. For multi-arm designs, it is the per-comparison (combined) sample size. For enrichment designs, it is the (combined) sample size for the considered sub-population.
allocationRatioPlanned	The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to

the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.

main — The main title, default is "Dataset".

xlab — The x-axis label, default is "Stage".

ylab — The y-axis label.

legendTitle — The legend title, default is "".

palette — The palette, default is "Set1".

legendPosition — The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- -1: no legend will be shown
- NA: the algorithm tries to find a suitable position
- 0: legend position outside plot
- 1: legend position left top
- 2: legend position left center
- 3: legend position left bottom
- 4: legend position right top
- 5: legend position right center
- 6: legend position right bottom

showSource — Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

grid — An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

The conditional power is calculated only if effect size and sample size is specified.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
design <- getDesignGroupSequential(kMax = 2)

dataExample <- getDataset(
    n = c(20, 30),
    means = c(50, 51),
    stDevs = c(130, 140)
)

result <- getAnalysisResults(design = design,
    dataInput = dataExample, thetaH0 = 20,
    nPlanned = c(30), thetaH1 = 1.5, stage = 1)

if (require(ggplot2)) plot(result, thetaRange = c(0, 100))

## End(Not run)
```

---

plot.Dataset *Dataset Plotting*

Description

Plots a dataset.

Usage

```
## S3 method for class 'Dataset'
plot(
  x,
  y,
  ...,
  main = "Dataset",
  xlab = "Stage",
  ylab = NA_character_,
  legendTitle = "Group",
  palette = "Set1",
  showSource = FALSE,
  plotSettings = NULL
)
```

Arguments

x	The Dataset object to plot.
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).

...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
main	The main title, default is "Dataset".
xlab	The x-axis label, default is "Stage".
ylab	The y-axis label.
legendTitle	The legend title, default is "Group".
palette	The palette, default is "Set1".
showSource	Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values: • "commands": returns a character vector with plot commands • "axes": returns a list with the axes definitions • "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs) • "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs) Note: no plot object will be returned if showSource is a character.
plotSettings	An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot all kinds of datasets.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
# Plot a dataset of means
dataExample <- getDataset(
    n1 = c(22, 11, 22, 11),
    n2 = c(22, 13, 22, 13),
    means1 = c(1, 1.1, 1, 1),
    means2 = c(1.4, 1.5, 3, 2.5),
    stDevs1 = c(1, 2, 2, 1.3),
    stDevs2 = c(1, 2, 2, 1.3)
)
if (require(ggplot2)) plot(dataExample, main = "Comparison of Means")

# Plot a dataset of rates
dataExample <- getDataset(
    n1 = c(8, 10, 9, 11),
    n2 = c(11, 13, 12, 13),
    events1 = c(3, 5, 5, 6),
    events2 = c(8, 10, 12, 12)
)
if (require(ggplot2)) plot(dataExample, main = "Comparison of Rates")

## End(Not run)
```

plot.EventProbabilities

Event Probabilities Plotting

Description

Plots an object that inherits from class EventProbabilities.

Usage

```
## S3 method for class 'EventProbabilities'
plot(
  x,
  y,
  ...,
  allocationRatioPlanned = x$allocationRatioPlanned,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = 1L,
  legendTitle = NA_character_,
  palette = "Set1",
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  plotSettings = NULL
)
```

Arguments

x	The object that inherits from EventProbabilities.
y	An optional object that inherits from NumberOfSubjects.
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length kMax, the number of stages. It can be a vector of length kMax, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally allocationRatioPlanned is treated as a vector of length kMax, not a scalar.
main	The main title.
xlab	The x-axis label.
ylab	The y-axis label.
type	The plot type (default = 1). Note that at the moment only one type is available.
legendTitle	The legend title, default is "".

`palette` The palette, default is `"Set1"`.

`plotPointsEnabled`
Logical. If `TRUE`, additional points will be plotted.

`legendPosition` The position of the legend. By default (`NA_integer_`) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- `-1`: no legend will be shown
- `NA`: the algorithm tries to find a suitable position
- `0`: legend position outside plot
- `1`: legend position left top
- `2`: legend position left center
- `3`: legend position left bottom
- `4`: legend position right top
- `5`: legend position right center
- `6`: legend position right bottom

`showSource` Logical. If `TRUE`, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R `plot` function. Alternatively `showSource` can be defined as one of the following character values:

- `"commands"`: returns a character vector with plot commands
- `"axes"`: returns a list with the axes definitions
- `"test"`: all plot commands will be validated with `eval(parse())` and returned as character vector (function does not stop if an error occurs)
- `"validate"`: all plot commands will be validated with `eval(parse())` and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if `showSource` is a character.

`plotSettings` An object of class `PlotSettings` created by `getPlotSettings()`.

Details

Generic function to plot an event probabilities object.

Generic function to plot an event probabilities object.

Value

Returns a `ggplot2` object.

`plot.NumberOfSubjects` *Number Of Subjects Plotting*

Description

Plots an object that inherits from class `NumberOfSubjects`.

Usage

```
## S3 method for class 'NumberOfSubjects'
plot(
  x,
  y,
  ...,
  allocationRatioPlanned = NA_real_,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = 1L,
  legendTitle = NA_character_,
  palette = "Set1",
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  plotSettings = NULL
)
```

Arguments

x	The object that inherits from NumberOfSubjects.
y	An optional object that inherits from EventProbabilities.
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
allocationRatioPlanned	The planned allocation ratio n1 / n2 for a two treatment groups design, default is 1. Will be ignored if y is undefined.
main	The main title.
xlab	The x-axis label.
ylab	The y-axis label.
type	The plot type (default = 1). Note that at the moment only one type is available.
legendTitle	The legend title, default is "".
palette	The palette, default is "Set1".
plotPointsEnabled	Logical. If TRUE, additional points will be plotted.
legendPosition	The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually: • -1: no legend will be shown • NA: the algorithm tries to find a suitable position • 0: legend position outside plot • 1: legend position left top • 2: legend position left center • 3: legend position left bottom • 4: legend position right top • 5: legend position right center

- 6: legend position right bottom

showSource — Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot an "number of subjects" object.

Generic function to plot a "number of subjects" object.

Value

Returns a ggplot2 object.

---

plot.ParameterSet *Parameter Set Plotting*

Description

Plots an object that inherits from class ParameterSet.

Usage

```
## S3 method for class 'ParameterSet'
plot(
  x,
  y,
  ...,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = 1L,
  palette = "Set1",
  legendPosition = NA_integer_,
  showSource = FALSE,
  plotSettings = NULL
)
```

Arguments

x	The object that inherits from ParameterSet.
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
main	The main title.
xlab	The x-axis label.
ylab	The y-axis label.
type	The plot type (default = 1).
palette	The palette, default is "Set1".
legendPosition	The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually: • -1: no legend will be shown • NA: the algorithm tries to find a suitable position • 0: legend position outside plot • 1: legend position left top • 2: legend position left center • 3: legend position left bottom • 4: legend position right top • 5: legend position right center • 6: legend position right bottom
showSource	Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values: • "commands": returns a character vector with plot commands • "axes": returns a list with the axes definitions • "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs) • "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs) Note: no plot object will be returned if showSource is a character.
plotSettings	An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot a parameter set.

Value

Returns a ggplot2 object.

`plot.SimulationResults` *Simulation Results Plotting*

Description

Plots simulation results.

Usage

```
## S3 method for class 'SimulationResults'
plot(
  x,
  y,
  ...,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = NA_integer_,
  palette = "Set1",
  theta = seq(-1, 1, 0.01),
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  grid = 1,
  plotSettings = NULL
)
```

Arguments

`x`	The simulation results, obtained from `getSimulationSurvival()`.
`y`	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
`...`	Optional plot arguments. At the moment `xlim` and `ylim` are implemented for changing x or y axis limits without dropping data observations.
`main`	The main title.
`xlab`	The x-axis label.
`ylab`	The y-axis label.
`type`	The plot type (default = `1`). The following plot types are available: • `1`: creates a 'Overall Success' plot (multi-arm and enrichment only) • `2`: creates a 'Success per Stage' plot (multi-arm and enrichment only) • `3`: creates a 'Selected Arms per Stage' plot (multi-arm and enrichment only) • `4`: creates a 'Reject per Stage' or 'Rejected Arms per Stage' plot • `5`: creates a 'Overall Power and Early Stopping' plot • `6`: creates a 'Expected Number of Subjects and Power / Early Stop' or 'Expected Number of Events and Power / Early Stop' plot • `7`: creates an 'Overall Power' plot

- 8: creates an 'Overall Early Stopping' plot
- 9: creates an 'Expected Sample Size' or 'Expected Number of Events' plot
- 10: creates a 'Study Duration' plot (non-multi-arm and non-enrichment survival only)
- 11: creates an 'Expected Number of Subjects' plot (non-multi-arm and non-enrichment survival only)
- 12: creates an 'Analysis Times' plot (non-multi-arm and non-enrichment survival only)
- 13: creates a 'Cumulative Distribution Function' plot (non-multi-arm and non-enrichment survival only)
- 14: creates a 'Survival Function' plot (non-multi-arm and non-enrichment survival only)
- "all": creates all available plots and returns it as a grid plot or list

palette — The palette, default is "Set1".

theta — A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.

plotPointsEnabled — Logical. If TRUE, additional points will be plotted.

legendPosition — The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- -1: no legend will be shown
- NA: the algorithm tries to find a suitable position
- 0: legend position outside plot
- 1: legend position left top
- 2: legend position left center
- 3: legend position left bottom
- 4: legend position right top
- 5: legend position right center
- 6: legend position right bottom

showSource — Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

grid — An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot all kinds of simulation results.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
results <- getSimulationMeans(
    alternative = 0:4, stDev = 5,
    plannedSubjects = 40, maxNumberOfIterations = 1000
)
plot(results, type = 5)

## End(Not run)
```

---

plot.StageResults *Stage Results Plotting*

Description

Plots the conditional power together with the likelihood function.

Usage

```
## S3 method for class 'StageResults'
plot(
  x,
  y,
  ...,
  type = 1L,
  nPlanned,
  allocationRatioPlanned = 1,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  legendTitle = NA_character_,
  palette = "Set1",
  legendPosition = NA_integer_,
  showSource = FALSE,
  plotSettings = NULL
)
```

Arguments

x	The stage results at given stage, obtained from getStageResults() or getAnalysisResults().
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).

`...` Optional plot arguments. Furthermore the following arguments can be defined:

- `thetaRange`: A range of assumed effect sizes if testing means or a survival design was specified. Additionally, if testing means was selected, an assumed standard deviation can be specified (default is 1).
- `piTreatmentRange`: A range of assumed rates pi1 to calculate the conditional power. Additionally, if a two-sample comparison was selected, pi2 can be specified (default is the value from `getAnalysisResults()`).
- `directionUpper`: Specifies the direction of the alternative, only applicable for one-sided testing; default is `TRUE` which means that larger values of the test statistics yield smaller p-values.
- `thetaH0`: The null hypothesis value, default is 0 for the normal and the binary case, it is 1 for the survival case. For testing a rate in one sample, a value thetaH0 in (0,1) has to be specified for defining the null hypothesis H0: pi = thetaH0.

`type` The plot type (default = 1). Note that at the moment only one type (the conditional power plot) is available.

`nPlanned` The additional (i.e., "new" and not cumulative) sample size planned for each of the subsequent stages. The argument must be a vector with length equal to the number of remaining stages and contain the combined sample size from both treatment groups if two groups are considered. For survival outcomes, it should contain the planned number of additional events. For multi-arm designs, it is the per-comparison (combined) sample size. For enrichment designs, it is the (combined) sample size for the considered sub-population.

`allocationRatioPlanned`
The planned allocation ratio `n1 / n2` for a two treatment groups design, default is `1`. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. For simulating means and rates for a two treatment groups design, it can be a vector of length `kMax`, the number of stages. It can be a vector of length `kMax`, too, for multi-arm and enrichment designs. In these cases, a change of allocating subjects to treatment groups over the stages can be assessed. Note that internally `allocationRatioPlanned` is treated as a vector of length `kMax`, not a scalar.

`main` The main title.

`xlab` The x-axis label.

`ylab` The y-axis label.

`legendTitle` The legend title.

`palette` The palette, default is `"Set1"`.

`legendPosition` The position of the legend. By default (`NA_integer_`) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- `-1`: no legend will be shown
- `NA`: the algorithm tries to find a suitable position
- `0`: legend position outside plot
- `1`: legend position left top
- `2`: legend position left center
- `3`: legend position left bottom
- `4`: legend position right top
- `5`: legend position right center

- 6: legend position right bottom

showSource — Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot all kinds of stage results. The conditional power is calculated only if effect size and sample size is specified.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
design <- getDesignGroupSequential(
    kMax = 4, alpha = 0.025,
    informationRates = c(0.2, 0.5, 0.8, 1),
    typeOfDesign = "WT", deltaWT = 0.25
)
dataExample <- getDataset(
    n = c(20, 30, 30),
    means = c(50, 51, 55),
    stDevs = c(130, 140, 120)
)
stageResults <- getStageResults(design, dataExample, thetaH0 = 20)
if (require(ggplot2)) plot(stageResults, nPlanned = c(30), thetaRange = c(0, 100))

## End(Not run)
```

---

plot.SummaryFactory *Summary Factory Plotting*

Description

Plots a summary factory.

Usage

```
## S3 method for class 'SummaryFactory'
plot(x, y, ..., showSummary = FALSE)
```

Arguments

x	The summary factory object.
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
showSummary	Show the summary before creating the plot output, default is FALSE.

Details

Generic function to plot all kinds of summary factories.

Value

Returns a ggplot2 object.

---

plot.TrialDesign *Trial Design Plotting*

Description

Plots a trial design.

Usage

```
## S3 method for class 'TrialDesign'
plot(
  x,
  y,
  ...,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = 1L,
  palette = "Set1",
  theta = seq(-1, 1, 0.01),
  nMax = NA_integer_,
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  grid = 1,
  plotSettings = NULL
)

## S3 method for class 'TrialDesignCharacteristics'
plot(x, y, ..., type = 1L, grid = 1)
```

Arguments

x	The trial design, obtained from getDesignGroupSequential(), getDesignInverseNormal() or getDesignFisher().
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
main	The main title.
xlab	The x-axis label.
ylab	The y-axis label.
type	The plot type (default = 1). The following plot types are available: • 1: creates a 'Boundaries' plot • 3: creates a 'Stage Levels' plot • 4: creates a 'Error Spending' plot • 5: creates a 'Power and Early Stopping' plot • 6: creates an 'Average Sample Size and Power / Early Stop' plot • 7: creates an 'Power' plot • 8: creates an 'Early Stopping' plot • 9: creates an 'Average Sample Size' plot • "all": creates all available plots and returns it as a grid plot or list
palette	The palette, default is "Set1".
theta	A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.
nMax	The maximum sample size. Must be a positive integer of length 1.
plotPointsEnabled	Logical. If TRUE, additional points will be plotted.
legendPosition	The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually: • -1: no legend will be shown • NA: the algorithm tries to find a suitable position • 0: legend position outside plot • 1: legend position left top • 2: legend position left center • 3: legend position left bottom • 4: legend position right top • 5: legend position right center • 6: legend position right bottom
showSource	Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values: • "commands": returns a character vector with plot commands

- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

grid — An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot a trial design.

Generic function to plot a trial design.

Note that nMax is not an argument that it passed to ggplot2. Rather, the underlying calculations (e.g. power for different theta's or average sample size) are based on calls to function getPowerAndAverageSampleNumber() which has argument nMax. I.e., nMax is not an argument to ggplot2 but to getPowerAndAverageSampleNumber() which is called prior to plotting.

Value

Returns a ggplot2 object.

See Also

plot() to compare different designs or design parameters visual.

Examples

```
## Not run:
design <- getDesignInverseNormal(
    kMax = 3, alpha = 0.025,
    typeOfDesign = "asKD", gammaA = 2,
    informationRates = c(0.2, 0.7, 1),
    typeBetaSpending = "bsOF"
)
if (require(ggplot2)) {
    plot(design) # default: type = 1
}

## End(Not run)
```

`plot.TrialDesignPlan` *Trial Design Plan Plotting*

Description

Plots a trial design plan.

Usage

```
## S3 method for class 'TrialDesignPlan'
plot(
  x,
  y,
  ...,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  type = NA_integer_,
  palette = "Set1",
  theta = NA_real_,
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  grid = 1,
  plotSettings = NULL
)
```

Arguments

x	The trial design plan, obtained from `getSampleSizeMeans()`, `getSampleSizeRates()`, `getSampleSizeSurvival()`, `getSampleSizeCounts()`, `getPowerMeans()`, `getPowerRates()` or `getPowerSurvival()` or `getPowerCounts()`.
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. At the moment `xlim` and `ylim` are implemented for changing x or y axis limits without dropping data observations.
main	The main title.
xlab	The x-axis label.
ylab	The y-axis label.
type	The plot type (default = 1). The following plot types are available: • 1: creates a 'Boundaries' plot • 2: creates a 'Boundaries Effect Scale' plot • 3: creates a 'Boundaries p Values Scale' plot

- 4: creates a 'Error Spending' plot
- 5: creates a 'Sample Size' or 'Overall Power and Early Stopping' plot
- 6: creates a 'Number of Events' or 'Sample Size' plot
- 7: creates an 'Overall Power' plot
- 8: creates an 'Overall Early Stopping' plot
- 9: creates an 'Expected Number of Events' or 'Expected Sample Size' plot
- 10: creates a 'Study Duration' plot
- 11: creates an 'Expected Number of Subjects' plot
- 12: creates an 'Analysis Times' plot
- 13: creates a 'Cumulative Distribution Function' plot
- 14: creates a 'Survival Function' plot
- "all": creates all available plots and returns it as a grid plot or list

`palette` The palette, default is "Set1".

`theta` A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.

`plotPointsEnabled`
Logical. If TRUE, additional points will be plotted.

`legendPosition` The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- -1: no legend will be shown
- NA: the algorithm tries to find a suitable position
- 0: legend position outside plot
- 1: legend position left top
- 2: legend position left center
- 3: legend position left bottom
- 4: legend position right top
- 5: legend position right center
- 6: legend position right bottom

`showSource` Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

`grid` An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

`plotSettings` An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot all kinds of trial design plans.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
if (require(ggplot2)) plot(getSampleSizeMeans())

## End(Not run)
```

---

plot.TrialDesignSet *Trial Design Set Plotting*

Description

Plots a trial design set.

Usage

```
## S3 method for class 'TrialDesignSet'
plot(
  x,
  y,
  ...,
  type = 1L,
  main = NA_character_,
  xlab = NA_character_,
  ylab = NA_character_,
  palette = "Set1",
  theta = seq(-1, 1, 0.02),
  nMax = NA_integer_,
  plotPointsEnabled = NA,
  legendPosition = NA_integer_,
  showSource = FALSE,
  grid = 1,
  plotSettings = NULL
)
```

Arguments

x	The trial design set, obtained from getDesignSet().
y	Not available for this kind of plot (is only defined to be compatible to the generic plot function).
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
type	The plot type (default = 1). The following plot types are available:

- 1: creates a 'Boundaries' plot
- 3: creates a 'Stage Levels' plot
- 4: creates a 'Error Spending' plot
- 5: creates a 'Power and Early Stopping' plot
- 6: creates an 'Average Sample Size and Power / Early Stop' plot
- 7: creates an 'Power' plot
- 8: creates an 'Early Stopping' plot
- 9: creates an 'Average Sample Size' plot
- "all": creates all available plots and returns it as a grid plot or list

main — The main title.

xlab — The x-axis label.

ylab — The y-axis label.

palette — The palette, default is "Set1".

theta — A vector of standardized effect sizes (theta values), default is a sequence from -1 to 1.

nMax — The maximum sample size. Must be a positive integer of length 1.

plotPointsEnabled — Logical. If TRUE, additional points will be plotted.

legendPosition — The position of the legend. By default (NA_integer_) the algorithm tries to find a suitable position. Choose one of the following values to specify the position manually:

- -1: no legend will be shown
- NA: the algorithm tries to find a suitable position
- 0: legend position outside plot
- 1: legend position left top
- 2: legend position left center
- 3: legend position left bottom
- 4: legend position right top
- 5: legend position right center
- 6: legend position right bottom

showSource — Logical. If TRUE, the parameter names of the object will be printed which were used to create the plot; that may be, e.g., useful to check the values or to create own plots with the base R plot function. Alternatively showSource can be defined as one of the following character values:

- "commands": returns a character vector with plot commands
- "axes": returns a list with the axes definitions
- "test": all plot commands will be validated with eval(parse()) and returned as character vector (function does not stop if an error occurs)
- "validate": all plot commands will be validated with eval(parse()) and returned as character vector (function stops if an error occurs)

Note: no plot object will be returned if showSource is a character.

grid — An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

plotSettings — An object of class PlotSettings created by getPlotSettings().

Details

Generic function to plot a trial design set. Is, e.g., useful to compare different designs or design parameters visual.

Value

Returns a ggplot2 object.

Examples

```
## Not run:
design <- getDesignInverseNormal(
    kMax = 3, alpha = 0.025,
    typeOfDesign = "asKD", gammaA = 2,
    informationRates = c(0.2, 0.7, 1), typeBetaSpending = "bsOF"
)

# Create a set of designs based on the master design defined above
# and varied parameter 'gammaA'
designSet <- getDesignSet(design = design, gammaA = 4)

if (require(ggplot2)) plot(designSet, type = 1, legendPosition = 6)

## End(Not run)
```

---

plot.TrialDesignSummaries
Plot Trial Design Summaries

Description

Generic function to plot a TrialDesignSummaries object.

Usage

```
## S3 method for class 'TrialDesignSummaries'
plot(x, ..., type = 1L, grid = 1)
```

Arguments

x	a TrialDesignSummaries object to plot.
...	further arguments passed to or from other methods.
type	The plot type (default = 1). The following plot types are available: • 1: creates a 'Boundaries' plot • 3: creates a 'Stage Levels' plot • 4: creates a 'Error Spending' plot • 5: creates a 'Power and Early Stopping' plot • 6: creates an 'Average Sample Size and Power / Early Stop' plot • 7: creates an 'Power' plot

- 8: creates an 'Early Stopping' plot
- 9: creates an 'Average Sample Size' plot
- "all": creates all available plots and returns it as a grid plot or list

grid — An integer value specifying the output of multiple plots. By default (1) a list of ggplot objects will be returned. If a grid value > 1 was specified, a grid plot will be returned if the number of plots is <= specified grid value; a list of ggplot objects will be returned otherwise. If grid = 0 is specified, all plots will be created using print command and a list of ggplot objects will be returned invisible. Note that one of the following packages must be installed to create a grid plot: 'ggpubr', 'gridExtra', or 'cowplot'.

PlotSettings *Plot Settings*

Description

Class for plot settings.

Details

Collects typical plot settings in an object.

Fields

lineSize The line size.

pointSize The point size.

pointColor The point color, e.g., "red" or "blue".

mainTitleFontSize The main tile font size.

axesTextFontSize The text font size.

legendFontSize The legend font size.

scalingFactor The scaling factor.

---

plotTypes *Get Available Plot Types*

Description

Function to identify the available plot types of an object.

Usage

```
plotTypes(
  obj,
  output = c("numeric", "caption", "numcap", "capnum"),
  numberInCaptionEnabled = FALSE
)

getAvailablePlotTypes(
  obj,
  output = c("numeric", "caption", "numcap", "capnum"),
  numberInCaptionEnabled = FALSE
)
```

Arguments

obj	The object for which the plot types shall be identified, e.g. produced by getDesignGroupSequential or getSampleSizeMeans().
output	The output type. Can be one of c("numeric", "caption", "numcap", "capnum").
numberInCaptionEnabled	If TRUE, the number will be added to the caption, default is FALSE.

Details

plotTypes and getAvailablePlotTypes() are equivalent, i.e., plotTypes is a short form of getAvailablePlotTypes().

output:

1. numeric: numeric output
2. caption: caption as character output
3. numcap: list with number and caption
4. capnum: list with caption and number

Value

Returns a list if option is either capnum or numcap or returns a vector that is of character type for option=caption or of numeric type for option=numeric.

Examples

```
## Not run:
design <- getDesignInverseNormal(kMax = 2)
getAvailablePlotTypes(design, "numeric")
plotTypes(design, "caption")
getAvailablePlotTypes(design, "numcap")
plotTypes(design, "capnum")

## End(Not run)
```

PowerAndAverageSampleNumberResult *Power and Average Sample Number Result*

Description

Class for power and average sample number (ASN) results.

Details

This object cannot be created directly; use `getPowerAndAverageSampleNumber()` with suitable arguments to create it.

Fields

`nMax` The maximum sample size. Is a numeric vector of length 1 containing a whole number.

`theta` A vector of standardized effect sizes (theta values). Is a numeric vector.

`averageSampleNumber` The average sample number calculated for each value of `theta` or `nMax`, if the specified maximum sample size would be exceeded. Is a numeric vector.

`calculatedPower` The calculated power for the given scenario.

`overallEarlyStop` The overall early stopping probability. Is a numeric vector.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`overallReject` The overall rejection probability. Is a numeric vector.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`overallFutility` The overall stopping for futility probability. Is a numeric vector.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

print.Dataset *Print Dataset Values*

Description

`print` prints its `Dataset` argument and returns it invisibly (via `invisible(x)`).

Usage

```
## S3 method for class 'Dataset'
print(
  x,
  ...,
  markdown = NA,
  output = c("list", "long", "wide", "r", "rComplete")
)
```

Arguments

x	A Dataset object.
...	Ensures that all arguments (startinc

print.InstallationQualificationResult *Print Installation Qualification Result*

Description

This function prints the details of an `InstallationQualificationResult` object in a user-friendly format.

Usage

```
## S3 method for class 'InstallationQualificationResult'
print(x, ...)
```

Arguments

`x`	An object of class `InstallationQualificationResult` containing the results of the installation qualification.
`...`	Additional arguments passed to or from other methods.

Details

The function displays the result message, followed by the parameters and their values. It skips parameters with `NULL` or `NA` values.

Value

This function does not return a value. It is called for its side effects of printing the result.

Examples

```
## Not run:
result <- testPackage()
print(result)

## End(Not run)
```

print.ParameterSet *Print Parameter Set Values*

Description

`print` prints its `ParameterSet` argument and returns it invisibly (via `invisible(x)`).

Usage

```
## S3 method for class 'ParameterSet'
print(x, ..., markdown = NA)
```

Arguments

x	The ParameterSet object to print.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
markdown	If TRUE, the object x will be printed using markdown syntax; normal representation will be used otherwise (default is FALSE)

Details

Prints the parameters and results of a parameter set.

---

print.SummaryFactory *Summary Factory Printing*

Description

Prints the result object stored inside a summary factory.

Usage

```
## S3 method for class 'SummaryFactory'
print(x, ..., markdown = NA, sep = NA_character_)
```

Arguments

x	The summary factory object.
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
markdown	If TRUE, the object x will be printed using markdown syntax; normal representation will be used otherwise (default is FALSE)
sep	The separator line between the summary and the print output, default is "\n\n-----\n\n".

Details

Generic function to print all kinds of summary factories.

---

print.TrialDesignCharacteristics

Trial Design Characteristics Printing

Description

Prints the design characteristics object.

Usage

```
## S3 method for class 'TrialDesignCharacteristics'
print(x, ..., markdown = NA, showDesign = TRUE)
```

Arguments

x	The trial design characteristics object.
...	Optional plot arguments. At the moment xlim and ylim are implemented for changing x or y axis limits without dropping data observations.
markdown	If TRUE, the object x will be printed using markdown syntax; normal representation will be used otherwise (default is FALSE)
showDesign	Show the design print output above the design characteristics, default is TRUE.

Details

Generic function to print all kinds of design characteristics.

---

print.TrialDesignSummaries *Print Trial Design Summaries*

Description

Generic function to print a TrialDesignSummaries object.

Usage

```
## S3 method for class 'TrialDesignSummaries'
print(x, ...)
```

Arguments

x	a TrialDesignSummaries object to print.
...	further arguments passed to or from other methods.

---

printCitation *Print Citation*

Description

How to cite rpact and R in publications.

Usage

```
printCitation(inclusiveR = TRUE, language = "en", markdown = NA)
```

Arguments

inclusiveR	If TRUE (default) the information on how to cite the base R system in publications will be added.
language	Language code to use for the output, default is "en".
markdown	If TRUE, the output will be created in Markdown.

Details

This function shows how to cite rpact and R (inclusiveR = TRUE) in publications.

Examples

```
printCitation()
```

---

rawDataTwoArmNormal *Raw Dataset Of A Two Arm Continuous Outcome With Covariates*

Description

An artificial dataset that was randomly generated with simulated normal data. The data set has six variables:

1. Subject id
2. Stage number
3. Group name
4. An example outcome in that we are interested in
5. The first covariate *gender*
6. The second covariate *covariate*

Usage

```
rawDataTwoArmNormal
```

Format

A data.frame object.

Details

See the vignette "Two-arm analysis for continuous data with covariates from raw data" to learn how to

- import raw data from a csv file,
- calculate estimated adjusted (marginal) means (EMMs, least-squares means) for a linear model, and
- perform two-arm interim analyses with these data.

You can use rawDataTwoArmNormal to reproduce the examples in the vignette.

rcmd *Get Object R Code*

Description

Returns the R source command of a result object.

Usage

```
rcmd(
  obj,
  ...,
  leadingArguments = NULL,
  includeDefaultParameters = FALSE,
  stringWrapParagraphWidth = 90,
  prefix = "",
  postfix = "",
  stringWrapPrefix = "",
  newArgumentValues = list(),
  tolerance = 1e-07,
  pipeOperator = c("auto", "none", "magrittr", "R"),
  output = c("vector", "cat", "test", "markdown", "internal"),
  explicitPrint = FALSE
)

getObjectRCode(
  obj,
  ...,
  leadingArguments = NULL,
  includeDefaultParameters = FALSE,
  stringWrapParagraphWidth = 90,
  prefix = "",
  postfix = "",
  stringWrapPrefix = "",
  newArgumentValues = list(),
  tolerance = 1e-07,
  pipeOperator = c("auto", "none", "magrittr", "R"),
  output = c("vector", "cat", "test", "markdown", "internal"),
  explicitPrint = FALSE
)
```

Arguments

obj	The result object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
leadingArguments	A character vector with arguments that shall be inserted at the beginning of the function command, e.g., `design = x`. Be careful with this option because the created R command may no longer be valid if used.

includeDefaultParameters
: If TRUE, default parameters will be included in all rpact commands; default is FALSE.

stringWrapParagraphWidth
: An integer value defining the number of characters after which a line break shall be inserted; set to NULL to insert no line breaks.

prefix
: A character string that shall be added to the beginning of the R command.

postfix
: A character string that shall be added to the end of the R command.

stringWrapPrefix
: A prefix character string that shall be added to each new line, typically some spaces.

newArgumentValues
: A named list with arguments that shall be renewed in the R command, e.g., newArgumentValues = list(informationRates = c(0.5, 1)).

tolerance
: The tolerance for defining a value as default.

pipeOperator
: The pipe operator to use in the R code, default is "none".

output
: The output format, default is a character "vector".

explicitPrint
: Show an explicit print command, default is FALSE.

Details

getObjectRCode() (short: rcmd()) recreates the R commands that result in the specified object obj. obj must be an instance of class ParameterSet.

Value

A character value or vector will be returned.

---

readDataset *Read Dataset*

Description

Reads a data file and returns it as dataset object.

Usage

```
readDataset(
  file,
  ...,
  header = TRUE,
  sep = ",",
  quote = "\"",
  dec = ".",
  fill = TRUE,
  comment.char = "",
  fileEncoding = "UTF-8"
)
```

Arguments

file	A CSV file (see read.table).
...	Further arguments to be passed to read.table.
header	A logical value indicating whether the file contains the names of the variables as its first line.
sep	The field separator character. Values on each line of the file are separated by this character. If sep = "," (the default for readDataset) the separator is a comma.
quote	The set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behavior on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.
dec	The character used in the file for decimal points.
fill	logical. If TRUE then in case the rows have unequal length, blank fields are implicitly added.
comment.char	character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.
fileEncoding	character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, the 'R Data Import/Export Manual' and 'Note'.

Details

readDataset is a wrapper function that uses read.table to read the CSV file into a data frame, transfers it from long to wide format with reshape and puts the data to getDataset().

Value

Returns a Dataset object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object,
- summary() to display a summary of the object,
- plot() to plot the object,
- as.data.frame() to coerce the object to a data.frame,
- as.matrix() to coerce the object to a matrix.

See Also

- readDatasets() for reading multiple datasets,
- writeDataset() for writing a single dataset,
- writeDatasets() for writing multiple datasets.

Examples

```
## Not run:
dataFileRates <- system.file("extdata",
    "dataset_rates.csv",
    package = "rpact"
)
if (dataFileRates != "") {
    datasetRates <- readDataset(dataFileRates)
    datasetRates
}

dataFileMeansMultiArm <- system.file("extdata",
    "dataset_means_multi-arm.csv",
    package = "rpact"
)
if (dataFileMeansMultiArm != "") {
    datasetMeansMultiArm <- readDataset(dataFileMeansMultiArm)
    datasetMeansMultiArm
}

dataFileRatesMultiArm <- system.file("extdata",
    "dataset_rates_multi-arm.csv",
    package = "rpact"
)
if (dataFileRatesMultiArm != "") {
    datasetRatesMultiArm <- readDataset(dataFileRatesMultiArm)
    datasetRatesMultiArm
}

dataFileSurvivalMultiArm <- system.file("extdata",
    "dataset_survival_multi-arm.csv",
    package = "rpact"
)
if (dataFileSurvivalMultiArm != "") {
    datasetSurvivalMultiArm <- readDataset(dataFileSurvivalMultiArm)
    datasetSurvivalMultiArm
}

## End(Not run)
```

`readDatasets` *Read Multiple Datasets*

Description

Reads a data file and returns it as a list of dataset objects.

Usage

```
readDatasets(
  file,
  ...,
  header = TRUE,
```

```
  sep = ",",
  quote = "\"",
  dec = ".",
  fill = TRUE,
  comment.char = "",
  fileEncoding = "UTF-8"
)
```

Arguments

file	A CSV file (see read.table).
...	Further arguments to be passed to read.table.
header	A logical value indicating whether the file contains the names of the variables as its first line.
sep	The field separator character. Values on each line of the file are separated by this character. If sep = "," (the default for readDatasets) the separator is a comma.
quote	The set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behavior on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.
dec	The character used in the file for decimal points.
fill	logical. If TRUE then in case the rows have unequal length, blank fields are implicitly added.
comment.char	character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.
fileEncoding	character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, the 'R Data Import/Export Manual' and 'Note'.

Details

Reads a file that was written by writeDatasets() before.

Value

Returns a list of Dataset objects.

See Also

- readDataset() for reading a single dataset,
- writeDatasets() for writing multiple datasets,
- writeDataset() for writing a single dataset.

Examples

```
## Not run:
dataFile <- system.file("extdata", "datasets_rates.csv", package = "rpact")
if (dataFile != "") {
    datasets <- readDatasets(dataFile)
    datasets
}
```

```
## End(Not run)
```

`readKeyValueFile` *Read a key-value file (KEY=VALUE) into a named list*

Description

Reads a human-editable key-value file in a widely used format (INI/.env-like): `KEY=VALUE`. Blank lines are ignored. Full-line comments starting with `#` or `;` are ignored. Inline comments are supported for unquoted values when preceded by whitespace, e.g., `KEY=123 # comment`.

Usage

```
readKeyValueFile(
  filePath,
  ...,
  inferTypes = TRUE,
  duplicateKeys = c("error", "last", "first"),
  safeKeyCheck = FALSE
)
```

Arguments

`filePath`	Path to the key-value file.
`...`	Currently unused.
`inferTypes`	Logical; if `TRUE`, attempts to convert values to logical/integer/numeric and `NA`. If `FALSE`, returns character values.
`duplicateKeys`	How to handle duplicate keys: `"error"` (default), `"last"` (keep last occurrence), or `"first"` (keep first occurrence).
`safeKeyCheck`	Logical; if `TRUE`, checks that keys only contain allowed characters. Set to `FALSE` (default) to allow arbitrary keys, but be aware that this may lead to issues when reading the file back, as keys with special characters may not be parsed correctly.

Details

Values can be quoted with double quotes. Escape sequences `\n`, `\r`, `\t`, `\\`, and `\"` are supported.

UTF-8 handling: The file is read as UTF-8 and all character values are normalized to UTF-8 via `enc2utf8()`.

Value

A named list with parsed values.

Examples

```
## Not run:
keyValueList <- list(
    STUDY_NAME = "Trial A",
    MAX_PATIENTS = 150L,
    THRESHOLD = 0.075,
    NOTES = "First phase\nSecond phase"
)
filePath <- tempfile(fileext = ".txt")
writeKeyValueFile(
    keyValueList = keyValueList,
    filePath = filePath,
    writeHeader = TRUE,
    sortKeys = TRUE,
    overwrite = TRUE
)
readKeyValueFile(filePath)

## End(Not run)
```

---

resetLogLevel *Reset Log Level*

Description

Resets the rpact log level.

Usage

```
resetLogLevel()
```

Details

This function resets the log level of the rpact internal log message system to the default value "PROGRESS".

See Also

- getLogLevel() for getting the current log level,
- setLogLevel() for setting the log level.

Examples

```
## Not run:
# reset log level to default value
resetLogLevel()

## End(Not run)
```

resetOptions *Reset Options*

Description

Resets the rpact options to their default values.

Usage

```
resetOptions(persist = TRUE)
```

Arguments

persist	A logical value indicating whether the reset options should be saved persistently. If TRUE, the options will be saved after resetting. Default is TRUE.

Details

This function resets all rpact options to their default values. If the persist parameter is set to TRUE, the reset options will be saved to a configuration file.

Value

Returns TRUE if the options were successfully reset, FALSE otherwise.

Examples

```
## Not run:
resetOptions()
resetOptions(persist = FALSE)

## End(Not run)
```

rpact *rpact - Confirmatory Adaptive Clinical Trial Design and Analysis*

Description

rpact (R Package for Adaptive Clinical Trials) is a comprehensive package that enables the design, simulation, and analysis of confirmatory adaptive group sequential designs. Particularly, the methods described in the recent monograph by Wassmer and Brannath (published by Springer, 2025) are implemented. It also comprises advanced methods for sample size calculations for fixed sample size designs incl., e.g., sample size calculation for survival trials with piecewise exponentially distributed survival times and staggered patients entry.

Details

rpact includes the classical group sequential designs (incl. user spending function approaches) where the sample sizes per stage (or the time points of interim analysis) cannot be changed in a data-driven way. Confirmatory adaptive designs explicitly allow for this under control of the Type I error rate. They are either based on the combination testing or the conditional rejection probability (CRP) principle. Both are available, for the former the inverse normal combination test and Fisher's combination test can be used.

Specific techniques of the adaptive methodology are also available, e.g., overall confidence intervals, overall p-values, and conditional and predictive power assessments. Simulations can be performed to assess the design characteristics of a (user-defined) sample size recalculation strategy. Designs are available for trials with continuous, binary, and survival endpoint.

For more information please visit www.rpact.org. If you are interested in professional services round about the package or need a comprehensive validation documentation to fulfill regulatory requirements please visit www.rpact.com.

rpact is developed by

- Gernot Wassmer (<gernot.wassmer@rpact.com>) and
- Friedrich Pahlke (<friedrich.pahlke@rpact.com>).

Author(s)

Gernot Wassmer, Friedrich Pahlke

References

Wassmer, G., Brannath, W. (2025) Group Sequential and Confirmatory Adaptive Designs in Clinical Trials (Springer Series in Pharmaceutical Statistics; doi:10.1007/9783031896699)

See Also

Useful links:

- https://www.rpact.org
- https://www.rpact.com
- https://docs.rpact.org
- https://github.com/rpact-com/rpact
- https://rpact-cloud.share.connect.posit.cloud
- Report bugs at https://github.com/rpact-com/rpact/issues

---

saveOptions *Save Options*

Description

Saves the current rpact options to a configuration file.

Usage

```
saveOptions()
```

Details

This function attempts to save the current rpact options to a configuration file located in the user's configuration directory. If the rappdirs package is not installed, the function will not perform any action. The options are saved in a YAML file.

Value

Returns TRUE if the options were successfully saved, FALSE otherwise.

Examples

```
## Not run:
saveOptions()

## End(Not run)
```

---

setLogLevel *Set Log Level*

Description

Sets the rpact log level.

Usage

```
setLogLevel(
  logLevel = c("PROGRESS", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "DISABLED")
)
```

Arguments

logLevel	The new log level to set. Can be one of "PROGRESS", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "DISABLED". Default is "PROGRESS".

Details

This function sets the log level of the rpact internal log message system. By default only calculation progress messages will be shown on the output console, particularly getAnalysisResults() shows this kind of messages. The output of these messages can be disabled by setting the log level to "DISABLED".

See Also

- getLogLevel() for getting the current log level,
- resetLogLevel() for resetting the log level to default.

Examples

```
## Not run:
# show debug messages
setLogLevel("DEBUG")

# disable all log messages
setLogLevel("DISABLED")

## End(Not run)
```

`setOutputFormat` *Set Output Format*

Description

With this function the format of the standard outputs of all `rpact` objects can be changed and set user defined respectively.

Usage

```
setOutputFormat(
  parameterName = NA_character_,
  ...,
  digits = NA_integer_,
  nsmall = NA_integer_,
  trimSingleZeros = NA,
  futilityProbabilityEnabled = NA,
  file = NA_character_,
  resetToDefault = FALSE,
  roundFunction = NA_character_,
  persist = TRUE
)
```

Arguments

`parameterName`	The name of the parameter whose output format shall be edited. Leave the default `NA_character_` if the output format of all parameters shall be edited.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`digits`	How many significant digits are to be used for a numeric value. The default, `NULL`, uses getOption("digits"). Allowed values are `0` <= `digits` <= `20`.
`nsmall`	The minimum number of digits to the right of the decimal point in formatting real numbers in non-scientific formats. Allowed values are `0` <= `nsmall` <= `20`.
`trimSingleZeros`	If `TRUE` zero values will be trimmed in the output, e.g., "0.00" will displayed as "0"
`futilityProbabilityEnabled`	If `TRUE` very small value (< 1e-09) will be displayed as "0", default is `FALSE`.

file	An optional file name of an existing text file that contains output format definitions (see Details for more information).
resetToDefault	If TRUE all output formats will be reset to default value. Note that other settings will be executed afterwards if specified, default is FALSE.
roundFunction	A character value that specifies the R base round function to use, default is NA_character_. Allowed values are "ceiling", "floor", "trunc", "round", "signif", and NA_character_.
persist	A logical value indicating whether the output format settings should be saved persistently. Default is TRUE.

Details

Output formats can be written to a text file (see getOutputFormat()). To load your personal output formats read a formerly saved file at the beginning of your work with rpact, e.g. execute setOutputFormat(file = "my_rpact_output_formats.txt").

Note that the parameterName must not match exactly, e.g., for p-values the following parameter names will be recognized amongst others:

1. p value
2. p.values
3. p-value
4. pValue
5. rpact.output.format.p.value

See Also

format for details on the function used internally to format the values.

Other output formats: getOutputFormat()

Examples

```
## Not run:
# show output format of p values
getOutputFormat("p.value")

# set new p value output format
setOutputFormat("p.value", digits = 5, nsmall = 5)

# show sample sizes as smallest integers not less than the not rounded values
setOutputFormat("sample size", digits = 0, nsmall = 0, roundFunction = "ceiling")
getSampleSizeMeans()

# show sample sizes as smallest integers not greater than the not rounded values
setOutputFormat("sample size", digits = 0, nsmall = 0, roundFunction = "floor")
getSampleSizeMeans()

# set new sample size output format without round function
setOutputFormat("sample size", digits = 2, nsmall = 2)
getSampleSizeMeans()

# reset sample size output format to default
setOutputFormat("sample size")
```

```
getSampleSizeMeans()
getOutputFormat("sample size")

## End(Not run)
```

`setupPackageTests` *Setup Package Tests*

Description

This function sets up the package tests by downloading the test files and copying them to the rpact installation directory.

Usage

```
setupPackageTests(token, secret)
```

Arguments

`token`	A character string representing the token for authentication.
`secret`	A character string representing the secret for authentication.

Details

The function first checks if the `rpact` package directory and its `tests` and `testthat` subdirectories exist. If they do not exist, it stops with an error. It then downloads the test files to a temporary directory and copies them to the `tests` directory of the `rpact` package. If all test files are copied successfully, it removes the default test file.

Value

The function returns `TRUE` if all test files are downloaded and copied successfully to the rpact installation directory; otherwise, it returns `FALSE`.

References

For more information, please visit: https://www.rpact.org/vignettes/utilities/rpact_installation_qualification/

SimulationResults *Class for Simulation Results*

Description

A class for simulation results.

Details

`SimulationResults` is the basic class for

- `SimulationResultsMeans`,
- `SimulationResultsRates`,
- `SimulationResultsSurvival`,
- `SimulationResultsCountData`,
- `SimulationResultsMultiArmMeans`,
- `SimulationResultsMultiArmRates`,
- `SimulationResultsMultiArmSurvival`,
- `SimulationResultsEnrichmentMeans`,
- `SimulationResultsEnrichmentRates`, and
- `SimulationResultsEnrichmentSurvival`.

Fields

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

SimulationResultsCountData *Class for Simulation Results Count Data*

Description

A class for simulation results count data.

Details

Use `getSimulationCounts()` to create an object of this type.

Fields

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`groups` The group numbers. Is a numeric vector.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`overallReject` The overall rejection probability. Is a numeric vector.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`SimulationResultsEnrichmentMeans`
Class for Simulation Results Enrichment Means

Description

A class for simulation results means in enrichment designs.

Details

Use `getSimulationEnrichmentMeans()` to create an object of this type.

Fields

`adaptations` Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length `kMax` minus 1.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`calcSubjectsFunction` An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerStage`.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effectList` The list of subsets, prevalences and effect sizes with columns and number of rows reflecting the different situations to be considered.

`effectMeasure` Criterion for treatment arm/population selection, either based on test statistic (`"testStatistic"`) or effect estimate (`"effectEstimate"`). Is a character vector of length 1.

`epsilonValue` Needs to be specified if `typeOfSelection = "epsilon"`. Is a numeric vector of length 1.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjectsPerStage` Determines the maximum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`minNumberOfSubjectsPerStage` Determines the minimum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`numberOfPopulations` The number of populations in an enrichment design. Is a numeric matrix.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`populations` The number of populations in an enrichment design. Is a numeric vector of length 1 containing a whole number.

`rejectAtLeastOne` The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

`rejectedPopulationsPerStage` The simulated number of rejected populations per stage.

`rValue` Needs to be specified if `typeOfSelection = "rBest"`. Is a numeric vector of length 1.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`selectedPopulations` The selected populations in enrichment designs.

`selectPopulationsFunction` An optional function that can be entered to define the way of how populations are selected.

`stDev` The standard deviation used for sample size and power calculation. Is a numeric vector of length 1.

`stDevH1` The standard deviation under which the conditional power or sample size recalculation is performed. Is a numeric vector of length 1.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

`successCriterion` Defines when the study is stopped for efficacy at interim. `"all"` stops the trial if the efficacy criterion has been fulfilled for all selected treatment arms/populations, `"atLeastOne"` stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim. Is a character vector of length 1.

`successPerStage` The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`threshold` The selection criterion: treatment arm/population is only selected if `effectMeasure` exceeds `threshold`. Either a single numeric value or a numeric vector of length `activeArms` referring to a separate threshold condition for each treatment arm.

`typeOfSelection` The way the treatment arms or populations are selected at interim. Is a character vector of length 1.

`SimulationResultsEnrichmentRates` *Class for Simulation Results Enrichment Rates*

Description

A class for simulation results rates in enrichment designs.

Details

Use `getSimulationEnrichmentRates()` to create an object of this type.

Fields

`adaptations` Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length `kMax` minus 1.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`calcSubjectsFunction` An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerStage`.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effectList` The list of subsets, prevalences and effect sizes with columns and number of rows reflecting the different situations to be considered.

`effectMeasure` Criterion for treatment arm/population selection, either based on test statistic (`"testStatistic"`) or effect estimate (`"effectEstimate"`). Is a character vector of length 1.

`epsilonValue` Needs to be specified if `typeOfSelection = "epsilon"`. Is a numeric vector of length 1.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjectsPerStage` Determines the maximum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`minNumberOfSubjectsPerStage` Determines the minimum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`numberOfPopulations` The number of populations in an enrichment design. Is a numeric matrix.

`piControlH1` The assumed probability in the reference group, for which the conditional power was calculated. Is a numeric vector of length 1 containing a value between 0 and 1.

`piTreatmentH1` The assumed probabilities in the active arm under which the sample size recalculation was performed and the conditional power was calculated.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`populations` The number of populations in an enrichment design. Is a numeric vector of length 1 containing a whole number.

`rejectAtLeastOne` The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

`rejectedPopulationsPerStage` The simulated number of rejected populations per stage.

`rValue` Needs to be specified if `typeOfSelection = "rBest"`. Is a numeric vector of length 1.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`selectedPopulations` The selected populations in enrichment designs.

`selectPopulationsFunction` An optional function that can be entered to define the way of how populations are selected.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

`successCriterion` Defines when the study is stopped for efficacy at interim. `"all"` stops the trial if the efficacy criterion has been fulfilled for all selected treatment arms/populations, `"atLeastOne"` stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim. Is a character vector of length 1.

`successPerStage` The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

`threshold` The selection criterion: treatment arm/population is only selected if `effectMeasure` exceeds `threshold`. Either a single numeric value or a numeric vector of length `activeArms` referring to a separate threshold condition for each treatment arm.

`typeOfSelection` The way the treatment arms or populations are selected at interim. Is a character vector of length 1.

`SimulationResultsEnrichmentSurvival`

Class for Simulation Results Enrichment Survival

Description

A class for simulation results survival in enrichment designs.

Details

Use `getSimulationEnrichmentSurvival()` to create an object of this type.

Fields

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`adaptations` Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length `kMax` minus 1.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`calcEventsFunction` An optional function that can be entered to define how event size is recalculated. By default, recalculation is performed with conditional power with specified `minNumberOfEventsPerStage` and `maxNumberOfEventsPerStage`.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`cumulativeEventsPerStage` The cumulative number of events per stage. Is a numeric matrix.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`dropoutRate1` The assumed drop-out rate in the treatment group. Is a numeric vector of length 1 containing a value between 0 and 1.

`dropoutRate2` The assumed drop-out rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

`dropoutTime` The assumed time for drop-out rates in the control and treatment group. Is a numeric vector of length 1.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effectList` The list of subsets, prevalences and effect sizes with columns and number of rows reflecting the different situations to be considered.

`effectMeasure` Criterion for treatment arm/population selection, either based on test statistic (`"testStatistic"`) or effect estimate (`"effectEstimate"`). Is a character vector of length 1.

`epsilonValue` Needs to be specified if `typeOfSelection = "epsilon"`. Is a numeric vector of length 1.

`eventsPerStage` Deprecated: use `singleEventsPerStage` or `cumulativeEventsPerStage` instead Is a numeric matrix.

`eventTime` The assumed time under which the event rates are calculated. Is a numeric vector of length 1.

`expectedNumberOfEvents` The expected number of events under specified alternative. Is a numeric vector.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`kappa` The shape of the Weibull distribution if `kappa!=1`. Is a numeric vector of length 1.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`maxNumberOfSubjectsPerStage` Determines the maximum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`minNumberOfSubjectsPerStage` Determines the minimum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`numberOfPopulations` The number of populations in an enrichment design. Is a numeric matrix.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`populations` The number of populations in an enrichment design. Is a numeric vector of length 1 containing a whole number.

`populationEventsPerStage` The cumulative number of events per stage Is a numeric matrix.

`rejectAtLeastOne` The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

`rejectedPopulationsPerStage` The simulated number of rejected populations per stage.

`rValue` Needs to be specified if `typeOfSelection = "rBest"`. Is a numeric vector of length 1.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`selectedPopulations` The selected populations in enrichment designs.

`selectPopulationsFunction` An optional function that can be entered to define the way of how populations are selected.

`singleNumberOfEventsPerStage` Deprecated: use `singleEventsPerArmAndStage` or `singleEventsPerSubsetAndS` instead

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

`studyDuration` The study duration for specified effect size. Is a positive numeric vector.

`successCriterion` Defines when the study is stopped for efficacy at interim. `"all"` stops the trial if the efficacy criterion has been fulfilled for all selected treatment arms/populations, `"atLeastOne"` stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim. Is a character vector of length 1.

`successPerStage` The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

threshold The selection criterion: treatment arm/population is only selected if effectMeasure exceeds threshold. Either a single numeric value or a numeric vector of length activeArms referring to a separate threshold condition for each treatment arm.

typeOfSelection The way the treatment arms or populations are selected at interim. Is a character vector of length 1.

SimulationResultsMeans

Class for Simulation Results Means

Description

A class for simulation results means.

Details

Use getSimulationMeans() to create an object of this type.

SimulationResultsMeans is the basic class for

- SimulationResultsMeans,
- SimulationResultsMultiArmMeans, and
- SimulationResultsEnrichmentMeans.

Fields

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

alternative The alternative hypothesis value(s) for testing means. Is a numeric vector.

calcSubjectsFunction An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerStage.

conditionalPower The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

conditionalPowerAchieved The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

effect The effect for randomly creating normally distributed responses. Is a numeric vector of length kMax.

expectedNumberOfSubjects The expected number of subjects under specified alternative.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

groups The group numbers. Is a numeric vector.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjectsPerStage` Determines the maximum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`meanRatio` Specifies if the sample size for one-sided testing of H0: `mu1/mu2 = thetaH0` has been calculated. Is a logical vector of length 1.

`minNumberOfSubjectsPerStage` Determines the minimum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`overallReject` The overall rejection probability. Is a numeric vector.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`stDev` The standard deviation used for sample size and power calculation. Is a numeric vector of length 1.

`stDevH1` The standard deviation under which the conditional power or sample size recalculation is performed. Is a numeric vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`SimulationResultsMultiArmMeans` *Class for Simulation Results Multi-Arm Means*

Description

A class for simulation results means in multi-arm designs.

Details

Use `getSimulationMultiArmMeans()` to create an object of this type.

Fields

`activeArms` The number of active treatment arms to be compared with control. Is a numeric vector of length 1 containing a whole number.

`adaptations` Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length `kMax` minus 1.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`calcSubjectsFunction` An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerStage`.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effectMatrix` The matrix of effect sizes with `activeArms` columns and number of rows reflecting the different situations to consider.

`effectMeasure` Criterion for treatment arm/population selection, either based on test statistic (`"testStatistic"`) or effect estimate (`"effectEstimate"`). Is a character vector of length 1.

`epsilonValue` Needs to be specified if `typeOfSelection = "epsilon"`. Is a numeric vector of length 1.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`gED50` The ED50 of the sigmoid Emax model. Only necessary if `typeOfShape = "sigmoidEmax"` has been specified. Is a numeric vector of length 1.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjectsPerStage` Determines the maximum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`minNumberOfSubjectsPerStage` Determines the minimum number of subjects per stage for data-driven sample size recalculation. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, is the minimum number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`muMaxVector` The range of effect sizes for the treatment group with highest response for `"linear"` and `"sigmoidEmax"` model. Is a numeric vector.

`numberOfActiveArms` The number of active arms in a multi-armed design. Is a numeric matrix.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`rejectAtLeastOne` The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

`rejectedArmsPerStage` The simulated number of rejected arms per stage.

`rValue` Needs to be specified if `typeOfSelection = "rBest"`. Is a numeric vector of length 1.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`selectArmsFunction` An optional function that can be entered to define how treatment arms are selected.

`selectedArms` The selected arms in multi-armed designs.

`slope` The slope of the sigmoid Emax model, if `typeOfShape = "sigmoidEmax"` Is a numeric vector of length 1.

`stDev` The standard deviation used for sample size and power calculation. Is a numeric vector of length 1.

`stDevH1` The standard deviation under which the conditional power or sample size recalculation is performed. Is a numeric vector of length 1.

`successCriterion` Defines when the study is stopped for efficacy at interim. `"all"` stops the trial if the efficacy criterion has been fulfilled for all selected treatment arms/populations, `"atLeastOne"` stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim. Is a character vector of length 1.

`successPerStage` The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`threshold` The selection criterion: treatment arm/population is only selected if `effectMeasure` exceeds `threshold`. Either a single numeric value or a numeric vector of length `activeArms` referring to a separate threshold condition for each treatment arm.

`typeOfSelection` The way the treatment arms or populations are selected at interim. Is a character vector of length 1.

`typeOfShape` The shape of the dose-response relationship over the treatment groups. Is a character vector of length 1.

`SimulationResultsMultiArmRates` *Class for Simulation Results Multi-Arm Rates*

Description

A class for simulation results rates in multi-arm designs.

Details

Use `getSimulationMultiArmRates()` to create an object of this type.

Fields

activeArms The number of active treatment arms to be compared with control. Is a numeric vector of length 1 containing a whole number.

adaptations Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length kMax minus 1.

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

calcSubjectsFunction An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified minNumberOfSubjectsPerStage and maxNumberOfSubjectsPerStage.

conditionalPower The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

conditionalPowerAchieved The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

effectMatrix The matrix of effect sizes with activeArms columns and number of rows reflecting the different situations to consider.

effectMeasure Criterion for treatment arm/population selection, either based on test statistic ("testStatistic") or effect estimate ("effectEstimate"). Is a character vector of length 1.

epsilonValue Needs to be specified if typeOfSelection = "epsilon". Is a numeric vector of length 1.

expectedNumberOfSubjects The expected number of subjects under specified alternative.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

gED50 The ED50 of the sigmoid Emax model. Only necessary if typeOfShape = "sigmoidEmax" has been specified. Is a numeric vector of length 1.

intersectionTest The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

iterations The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

maxNumberOfIterations The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

maxNumberOfSubjects The maximum number of subjects for power calculations. Is a numeric vector.

numberOfActiveArms The number of active arms in a multi-armed design. Is a numeric matrix.

piControl The assumed probability in the control arm for simulation and under which the sample size recalculation is performed. Is a numeric vector of length 1 containing a value between 0 and 1.

piControlH1 The assumed probability in the reference group, for which the conditional power was calculated. Is a numeric vector of length 1 containing a value between 0 and 1.

piH1 The assumed probability in the active treatment arm(s) under which the sample size recalculation is performed. Is a numeric vector of length 1 containing a value between 0 and 1.

piMaxVector The range of assumed probabilities for the treatment group with highest response for "linear" and "sigmoidEmax" model.

plannedSubjects Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length kMax containing whole numbers.

rejectAtLeastOne The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

rejectedArmsPerStage The simulated number of rejected arms per stage.

rValue Needs to be specified if typeOfSelection = "rBest". Is a numeric vector of length 1.

sampleSizes The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

seed The seed used for random number generation. Is a numeric vector of length 1.

selectArmsFunction An optional function that can be entered to define how treatment arms are selected.

selectedArms The selected arms in multi-armed designs.

slope The slope of the sigmoid Emax model, if typeOfShape = "sigmoidEmax" Is a numeric vector of length 1.

successCriterion Defines when the study is stopped for efficacy at interim. "all" stops the trial if the efficacy criterion has been fulfilled for all selected treatment arms/populations, "atLeastOne" stops if at least one of the selected treatment arms/populations is shown to be superior to control at interim. Is a character vector of length 1.

successPerStage The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

threshold The selection criterion: treatment arm/population is only selected if effectMeasure exceeds threshold. Either a single numeric value or a numeric vector of length activeArms referring to a separate threshold condition for each treatment arm.

typeOfSelection The way the treatment arms or populations are selected at interim. Is a character vector of length 1.

typeOfShape The shape of the dose-response relationship over the treatment groups. Is a character vector of length 1.

SimulationResultsMultiArmSurvival

Class for Simulation Results Multi-Arm Survival

Description

A class for simulation results survival in multi-arm designs.

Details

Use getSimulationMultiArmSurvival() to create an object of this type.

Fields

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`activeArms` The number of active treatment arms to be compared with control. Is a numeric vector of length 1 containing a whole number.

`adaptations` Indicates whether or not an adaptation takes place at interim k. Is a logical vector of length `kMax` minus 1.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`correlationComputation` If `"alternative"`, a correlation matrix according to Deng et al. (Biometrics, 2019) accounting for the respective alternative is used for simulating log-rank statistics in the many-to-one design. If `"null"`, a constant correlation matrix valid under the null hypothesis is used.

`cumulativeEventsPerStage` The cumulative number of events per stage. Is a numeric matrix.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`dropoutRate1` The assumed drop-out rate in the treatment group. Is a numeric vector of length 1 containing a value between 0 and 1.

`dropoutRate2` The assumed drop-out rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

`dropoutTime` The assumed time for drop-out rates in the control and treatment group. Is a numeric vector of length 1.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effectMatrix` The matrix of effect sizes with `activeArms` columns and number of rows reflecting the different situations to consider.

`epsilonValue` Needs to be specified if `typeOfSelection = "epsilon"`. Is a numeric vector of length 1.

`eventsPerStage` Deprecated: use `singleEventsPerStage` or `cumulativeEventsPerStage` instead Is a numeric matrix.

`eventTime` The assumed time under which the event rates are calculated. Is a numeric vector of length 1.

`expectedNumberOfEvents` The expected number of events under specified alternative. Is a numeric vector.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`gED50` The ED50 of the sigmoid Emax model. Only necessary if `typeOfShape = "sigmoidEmax"` has been specified. Is a numeric vector of length 1.

intersectionTest The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

iterations The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

kappa The shape of the Weibull distribution if kappa!=1. Is a numeric vector of length 1.

maxNumberOfEventsPerStage Determines the maximum number of events per stage for data-driven sample size recalculation. Is a numeric vector of length kMax containing whole numbers.

maxNumberOfIterations The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

maxNumberOfSubjects The maximum number of subjects for power calculations. Is a numeric vector.

minNumberOfEventsPerStage Determines the minimum number of events per stage for data-driven sample size recalculation. Is a numeric vector of length kMax containing whole numbers.

numberOfActiveArms The number of active arms in a multi-armed design. Is a numeric matrix.

omegaMaxVector The range of hazard ratios with highest response for "linear" and "sigmoidEmax" model. Is a numeric vector.

plannedEvents Determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, is the number of events for both treatment arms. For multi-arm designs, refers to the overall number of events for the selected arms plus control. Is a numeric vector of length kMax containing whole numbers.

rejectAtLeastOne The probability to reject at least one of the (multiple) hypotheses. Is a numeric vector.

rejectedArmsPerStage The simulated number of rejected arms per stage.

rValue Needs to be specified if typeOfSelection = "rBest". Is a numeric vector of length 1.

seed The seed used for random number generation. Is a numeric vector of length 1.

selectArmsFunction An optional function that can be entered to define how treatment arms are selected.

selectedArms The selected arms in multi-armed designs.

singleEventsPerArmAndStage The number of events per arm and stage that is used for the analysis.

singleEventsPerStage The single number of events per stage. Is a numeric matrix.

singleNumberOfEventsPerStage Deprecated: use singleEventsPerArmAndStage or singleEventsPerSubsetAndS instead

slope The slope of the sigmoid Emax model, if typeOfShape = "sigmoidEmax" Is a numeric vector of length 1.

studyDuration The study duration for specified effect size. Is a positive numeric vector.

successPerStage The simulated success probabilities per stage where success is defined by user. Is a numeric matrix.

threshold The selection criterion: treatment arm/population is only selected if effectMeasure exceeds threshold. Either a single numeric value or a numeric vector of length activeArms referring to a separate threshold condition for each treatment arm.

typeOfShape The shape of the dose-response relationship over the treatment groups. Is a character vector of length 1.

SimulationResultsRates

Class for Simulation Results Rates

Description

A class for simulation results rates.

Details

Use `getSimulationRates()` to create an object of this type.

`SimulationResultsRates` is the basic class for

- `SimulationResultsRates`,
- `SimulationResultsMultiArmRates`, and
- `SimulationResultsEnrichmentRates`.

Fields

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`calcSubjectsFunction` An optional function that can be entered to define how sample size is recalculated. By default, recalculation is performed with conditional power with specified `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerStage`.

`conditionalPower` The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

`conditionalPowerAchieved` The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`effect` The effect for randomly creating normally distributed responses. Is a numeric vector of length `kMax`.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`groups` The group numbers. Is a numeric vector.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`overallReject` The overall rejection probability. Is a numeric vector.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi1H1` The assumed probability in the active treatment group for two-group designs, or the assumed probability for a one treatment group design, for which the conditional power was calculated. Is a numeric vector of length 1 containing a value between 0 and 1.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`pi2H1` The assumed probability in the reference group for two-group designs, for which the conditional power was calculated. Is a numeric vector of length 1 containing a value between 0 and 1.

`plannedSubjects` Determines the number of cumulated (overall) subjects when the interim stages are planned. For two treatment arms, is the number of subjects for both treatment arms. For multi-arm designs, refers to the number of subjects per selected active arm. Is a numeric vector of length `kMax` containing whole numbers.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`riskRatio` Specifies if the sample size for one-sided testing of H0: `pi1 / pi2 = thetaH0` has been calculated. Is a logical vector of length 1.

`sampleSizes` The sample sizes for each group and stage. Is a numeric vector of length number of stages times number of groups containing whole numbers.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`SimulationResultsSurvival` *Class for Simulation Results Survival*

Description

A class for simulation results survival.

Details

Use `getSimulationSurvival()` to create an object of this type.

`SimulationResultsSurvival` is the basic class for

- `SimulationResultsSurvival`,
- `SimulationResultsMultiArmSurvival`, and
- `SimulationResultsEnrichmentSurvival`.

Fields

accrualIntensity The absolute accrual intensities. Is a numeric vector of length kMax.

accrualTime The assumed accrual time intervals for the study. Is a numeric vector.

allocation1 The number of subjects to be assigned to treatment 1 in subsequent order. Is a numeric vector of length 1 containing a whole number.

allocation2 The number of subjects to be assigned to treatment 2 in subsequent order. Is a numeric vector of length 1 containing a whole number.

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

calcEventsFunction An optional function that can be entered to define how event size is recalculated. By default, recalculation is performed with conditional power with specified minNumberOfEventsPerStage and maxNumberOfEventsPerStage.

conditionalPower The conditional power at each stage of the trial. Is a numeric vector of length 1 containing a value between 0 and 1.

conditionalPowerAchieved The calculated conditional power, under the assumption of observed or assumed effect sizes. Is a numeric matrix.

cumulativeEventsPerStage The cumulative number of events per stage. Is a numeric matrix.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

dropoutRate1 The assumed drop-out rate in the treatment group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutRate2 The assumed drop-out rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutTime The assumed time for drop-out rates in the control and treatment group. Is a numeric vector of length 1.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

eventsNotAchieved The simulated number of cases how often the number of events was not reached. Is a numeric matrix.

eventTime The assumed time under which the event rates are calculated. Is a numeric vector of length 1.

expectedNumberOfEvents The expected number of events under specified alternative. Is a numeric vector.

expectedNumberOfSubjects The expected number of subjects under specified alternative.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

hazardRatio The hazard ratios under consideration. Is a numeric vector of length kMax.

iterations The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

kappa The shape of the Weibull distribution if kappa!=1. Is a numeric vector of length 1.

lambda1 The assumed hazard rate in the treatment group. Is a numeric vector of length kMax.

lambda2 The assumed hazard rate in the reference group. Is a numeric vector of length 1.

`maxNumberOfEventsPerStage` Determines the maximum number of events per stage for data-driven sample size recalculation. Is a numeric vector of length `kMax` containing whole numbers.

`maxNumberOfIterations` The number of simulation iterations. Is a numeric vector of length 1 containing a whole number.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`median1` The assumed median survival time in the treatment group. Is a numeric vector.

`median2` The assumed median survival time in the reference group. Is a numeric vector of length 1.

`minNumberOfEventsPerStage` Determines the minimum number of events per stage for data-driven sample size recalculation. Is a numeric vector of length `kMax` containing whole numbers.

`numberOfSubjects` In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

`numberOfSubjects1` In simulation results data set: The number of subjects under consideration in treatment arm 1 when the interim analysis takes place.

`numberOfSubjects2` In simulation results data set: The number of subjects under consideration in treatment arm 2 when the interim analysis takes place.

`overallReject` The overall rejection probability. Is a numeric vector.

`pi1` The assumed event rate in the treatment group. Is a numeric vector of length `kMax` containing values between 0 and 1.

`pi2` The assumed event rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

`piecewiseSurvivalTime` The time intervals for the piecewise definition of the exponential survival time cumulative distribution function. Is a numeric vector.

`plannedEvents` Determines the number of cumulated (overall) events in survival designs when the interim stages are planned. For two treatment arms, is the number of events for both treatment arms. For multi-arm designs, refers to the overall number of events for the selected arms plus control. Is a numeric vector of length `kMax` containing whole numbers.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

`singleEventsPerStage` The single number of events per stage. Is a numeric matrix.

`studyDuration` The study duration for specified effect size. Is a positive numeric vector.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`thetaH1` The assumed effect under the alternative hypothesis. For survival designs, refers to the hazard ratio. Is a numeric vector.

`StageResults` *Basic Stage Results*

Description

Basic class for stage results.

Details

StageResults is the basic class for

- StageResultsMeans,
- StageResultsRates,
- StageResultsSurvival,
- StageResultsMultiArmMeans,
- StageResultsMultiArmRates,
- StageResultsMultiArmSurvival,
- StageResultsEnrichmentMeans,
- StageResultsEnrichmentRates, and
- StageResultsEnrichmentSurvival.

Fields

stages The stage numbers of the trial. Is a numeric vector of length kMax containing whole numbers.

testStatistics The stage-wise test statistics. Is a numeric vector of length kMax.

pValues The stage-wise p-values. Is a numeric vector of length kMax containing values between 0 and 1.

combInverseNormal The test statistics over stages for the inverse normal test. Is a numeric vector of length kMax.

combFisher The test statistics over stages for Fisher's combination test. Is a numeric vector of length kMax containing values between 0 and 1.

effectSizes The stage-wise effect sizes. Is a numeric vector of length kMax.

testActions The test decisions at each stage of the trial. Is a character vector of length kMax.

weightsFisher The weights for Fisher's combination test. Is a numeric vector of length kMax.

weightsInverseNormal The weights for the inverse normal statistic. Is a numeric vector of length kMax.

StageResultsEnrichmentMeans
Stage Results Enrichment Means

Description

Class for stage results of enrichment means data

Details

This object cannot be created directly; use getStageResults with suitable arguments to create the stage results of a dataset of enrichment means.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`direction` Specifies the direction of the alternative, is either "upper" or "lower". Only applicable for one-sided testing.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallStDevs` The overall, i.e., cumulative standard deviations. Is a numeric vector of length number of stages times number of groups.

`overallPooledStDevs` The overall pooled standard deviations. Is a numeric matrix.

`separatePValues` The p-values from the separate stages. Is a numeric matrix.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`singleStepAdjustedPValues` The adjusted p-value for testing multiple hypotheses per stage of the trial.

`stratifiedAnalysis` For enrichment designs, typically a stratified analysis should be chosen. When testing means and rates, a non-stratified analysis can be performed on overall data. For survival data, only a stratified analysis is possible. Is a logical vector of length 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

StageResultsEnrichmentRates
Stage Results Enrichment Rates

Description

Class for stage results of enrichment rates data.

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of enrichment rates.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

StageResultsEnrichmentSurvival
Stage Results Enrichment Survival

Description

Class for stage results of enrichment survival data.

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of enrichment survival.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`StageResultsMeans` *Stage Results of Means*

Description

Class for stage results of means.

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of means.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`direction` Specifies the direction of the alternative, is either "upper" or "lower". Only applicable for one-sided testing.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`equalVariances` Describes if the variances in two treatment groups are assumed to be the same. Is a logical vector of length 1.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`...` Names of `dataInput`.

`StageResultsMultiArmMeans`

Stage Results Multi Arm Means

Description

Class for stage results of multi arm means data

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of multi arm means.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`overallStDevs` The overall, i.e., cumulative standard deviations. Is a numeric vector of length number of stages times number of groups.

`overallPooledStDevs` The overall pooled standard deviations. Is a numeric matrix.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`separatePValues` The p-values from the separate stages. Is a numeric matrix.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`singleStepAdjustedPValues` The adjusted p-value for testing multiple hypotheses per stage of the trial.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`varianceOption` Defines the way to calculate the variance in multiple (i.e., >2) treatment arms or population enrichment designs when testing means. Available options for multiple arms: `"overallPooled"`, `"pairwisePooled"`, `"notPooled"`. Available options for enrichment designs: `"pooled"`, `"pooledFromFull"`, `"notPooled"`.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

StageResultsMultiArmRates

Stage Results Multi Arm Rates

Description

Class for stage results of multi arm rates data

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of multi arm rates.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`separatePValues` The p-values from the separate stages. Is a numeric matrix.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`singleStepAdjustedPValues` The adjusted p-value for testing multiple hypotheses per stage of the trial.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`StageResultsMultiArmSurvival`

Stage Results Multi Arm Survival

Description

Class for stage results of multi arm survival data

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of multi arm survival.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

combInverseNormal The test statistics over stages for the inverse normal test. Is a numeric vector of length kMax.

combFisher The test statistics over stages for Fisher's combination test. Is a numeric vector of length kMax containing values between 0 and 1.

effectSizes The stage-wise effect sizes. Is a numeric vector of length kMax.

testActions The test decisions at each stage of the trial. Is a character vector of length kMax.

weightsFisher The weights for Fisher's combination test. Is a numeric vector of length kMax.

weightsInverseNormal The weights for the inverse normal statistic. Is a numeric vector of length kMax.

combInverseNormal The test statistics over stages for the inverse normal test. Is a numeric vector of length kMax.

combFisher The test statistics over stages for Fisher's combination test. Is a numeric vector of length kMax containing values between 0 and 1.

overallTestStatistics The overall, i.e., cumulated test statistics. Is a numeric vector of length kMax.

overallPValues The overall, i.e., cumulated p-values. Is a numeric vector of length kMax containing values between 0 and 1.

testStatistics The stage-wise test statistics. Is a numeric vector of length kMax.

separatePValues The p-values from the separate stages. Is a numeric matrix.

effectSizes The stage-wise effect sizes. Is a numeric vector of length kMax.

singleStepAdjustedPValues The adjusted p-value for testing multiple hypotheses per stage of the trial.

intersectionTest The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

StageResultsRates *Stage Results of Rates*

Description

Class for stage results of rates.

Details

This object cannot be created directly; use getStageResults with suitable arguments to create the stage results of a dataset of rates.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`pValues` The stage-wise p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`direction` Specifies the direction of the alternative, is either "upper" or "lower". Only applicable for one-sided testing.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`...` Names of `dataInput`.

`StageResultsSurvival` *Stage Results of Survival Data*

Description

Class for stage results survival data.

Details

This object cannot be created directly; use `getStageResults` with suitable arguments to create the stage results of a dataset of survival data.

Fields

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`testStatistics` The stage-wise test statistics. Is a numeric vector of length `kMax`.

`overallTestStatistics` The overall, i.e., cumulated test statistics. Is a numeric vector of length `kMax`.

`separatePValues` The p-values from the separate stages. Is a numeric matrix.

`singleStepAdjustedPValues` The adjusted p-value for testing multiple hypotheses per stage of the trial.

`overallPValues` The overall, i.e., cumulated p-values. Is a numeric vector of length `kMax` containing values between 0 and 1.

`direction` Specifies the direction of the alternative, is either "upper" or "lower". Only applicable for one-sided testing.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`intersectionTest` The multiple test used for intersection hypotheses in closed systems of hypotheses. Is a character vector of length 1.

`combInverseNormal` The test statistics over stages for the inverse normal test. Is a numeric vector of length `kMax`.

`combFisher` The test statistics over stages for Fisher's combination test. Is a numeric vector of length `kMax` containing values between 0 and 1.

`effectSizes` The stage-wise effect sizes. Is a numeric vector of length `kMax`.

`testActions` The test decisions at each stage of the trial. Is a character vector of length `kMax`.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`weightsFisher` The weights for Fisher's combination test. Is a numeric vector of length `kMax`.

`weightsInverseNormal` The weights for the inverse normal statistic. Is a numeric vector of length `kMax`.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`...` Names of `dataInput`.

`summary.AnalysisResults`

Analysis Results Summary

Description

Displays a summary of `AnalysisResults` object.

Usage

```
## S3 method for class 'AnalysisResults'
summary(object, ..., type = 1, digits = NA_integer_)
```

Arguments

`object`	An `AnalysisResults` object.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`digits`	Defines how many digits are to be used for numeric values. Must be a positive integer of length 1.

Details

Summarizes the parameters and results of an analysis results object.

Value

Returns a SummaryFactory object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object

Summary options

The following options can be set globally:

1. rpact.summary.output.size: one of c("small", "medium", "large"); defines how many details will be included into the summary; default is "large", i.e., all available details are displayed.
2. rpact.summary.justify: one of c("right", "left", "centre"); shall the values be right-justified (the default), left-justified or centered.
3. rpact.summary.width: defines the maximum number of characters to be used per line (default is 83).
4. rpact.summary.intervalFormat: defines how intervals will be displayed in the summary, default is "[%s; %s]".
5. rpact.summary.digits: defines how many digits are to be used for numeric values (default is 3).
6. rpact.summary.digits.probs: defines how many digits are to be used for numeric values (default is one more than value of rpact.summary.digits, i.e., 4).
7. rpact.summary.trim.zeroes: if TRUE (default) zeroes will always displayed as "0", e.g. "0.000" will become "0".

Example: options("rpact.summary.intervalFormat" = "%s - %s")

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

---

summary.Dataset *Dataset Summary*

Description

Displays a summary of Dataset object.

Usage

```
## S3 method for class 'Dataset'
summary(object, ..., type = 1, digits = NA_integer_)
```

Arguments

object	A Dataset object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
digits	Defines how many digits are to be used for numeric values. Must be a positive integer of length 1.

Details

Summarizes the parameters and results of a dataset.

Value

Returns a SummaryFactory object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object

Summary options

The following options can be set globally:

1. rpact.summary.output.size: one of c("small", "medium", "large"); defines how many details will be included into the summary; default is "large", i.e., all available details are displayed.
2. rpact.summary.justify: one of c("right", "left", "centre"); shall the values be right-justified (the default), left-justified or centered.
3. rpact.summary.width: defines the maximum number of characters to be used per line (default is 83).
4. rpact.summary.intervalFormat: defines how intervals will be displayed in the summary, default is "[%s; %s]".
5. rpact.summary.digits: defines how many digits are to be used for numeric values (default is 3).
6. rpact.summary.digits.probs: defines how many digits are to be used for numeric values (default is one more than value of rpact.summary.digits, i.e., 4).
7. rpact.summary.trim.zeroes: if TRUE (default) zeroes will always displayed as "0", e.g. "0.000" will become "0".

Example: options("rpact.summary.intervalFormat" = "%s - %s")

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

summary.FutilityBounds *Summarize Futility Bounds*

Description

S3 summary method for objects of class `FutilityBounds`.

Usage

```
## S3 method for class 'FutilityBounds'
summary(object, ...)
```

Arguments

`object`	An object of class `FutilityBounds`.
`...`	Additional arguments (currently not used).

Details

Prints a categorized summary of futility bound parameters, including user-defined, derived, default, and generated values.

Examples

```
## Not run:
futilityBounds <- getFutilityBounds(
    design = getDesignInverseNormal(kMax = 2),
    sourceValue = 0.5,
    sourceScale = "condPowerAtObserved",
    targetScale = "zValue"
)
summary(futilityBounds)

## End(Not run)
```

summary.ParameterSet *Parameter Set Summary*

Description

Displays a summary of `ParameterSet` object.

Usage

```
## S3 method for class 'ParameterSet'
summary(
  object,
  ...,
  type = 1,
  digits = NA_integer_,
  output = c("all", "title", "overview", "body"),
  printObject = FALSE,
  sep = NA_character_
)
```

Arguments

object	A ParameterSet object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
digits	Defines how many digits are to be used for numeric values. Must be a positive integer of length 1.
output	The output parts, default is "all".
printObject	Show also the print output after the summary, default is FALSE.
sep	The separator line between the summary and the optional print output, default is "\n\n-----\n\n".

Details

Summarizes the parameters and results of a parameter set.

Value

Returns a SummaryFactory object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object

Summary options

The following options can be set globally:

1. rpact.summary.output.size: one of c("small", "medium", "large"); defines how many details will be included into the summary; default is "large", i.e., all available details are displayed.
2. rpact.summary.justify: one of c("right", "left", "centre"); shall the values be right-justified (the default), left-justified or centered.
3. rpact.summary.width: defines the maximum number of characters to be used per line (default is 83).
4. rpact.summary.intervalFormat: defines how intervals will be displayed in the summary, default is "[%s; %s]".
5. rpact.summary.digits: defines how many digits are to be used for numeric values (default is 3).

6. rpact.summary.digits.probs: defines how many digits are to be used for numeric values (default is one more than value of rpact.summary.digits, i.e., 4).
7. rpact.summary.trim.zeroes: if TRUE (default) zeroes will always displayed as "0", e.g. "0.000" will become "0".

Example: options("rpact.summary.intervalFormat" = "%s - %s")

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the rpact specific implementation of the generic. Note that you can use the R function methods to get all the methods of a generic and to identify the object specific name of it, e.g., use methods("plot") to get all the methods for the plot generic. There you can find, e.g., plot.AnalysisResults and obtain the specific help documentation linked above by typing ?plot.AnalysisResults.

---

summary.TrialDesignSet
Trial Design Set Summary

Description

Displays a summary of ParameterSet object.

Usage

```
## S3 method for class 'TrialDesignSet'
summary(object, ..., type = 1, digits = NA_integer_)
```

Arguments

object	A ParameterSet object.
...	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
digits	Defines how many digits are to be used for numeric values. Must be a positive integer of length 1.

Details

Summarizes the trial designs.

Value

Returns a SummaryFactory object. The following generics (R generic functions) are available for this result object:

- names() to obtain the field names,
- print() to print the object

Summary options

The following options can be set globally:

1. `rpact.summary.output.size`: one of `c("small", "medium", "large")`; defines how many details will be included into the summary; default is `"large"`, i.e., all available details are displayed.
2. `rpact.summary.justify`: one of `c("right", "left", "centre")`; shall the values be right-justified (the default), left-justified or centered.
3. `rpact.summary.width`: defines the maximum number of characters to be used per line (default is `83`).
4. `rpact.summary.intervalFormat`: defines how intervals will be displayed in the summary, default is `"[%s; %s]"`.
5. `rpact.summary.digits`: defines how many digits are to be used for numeric values (default is `3`).
6. `rpact.summary.digits.probs`: defines how many digits are to be used for numeric values (default is one more than value of `rpact.summary.digits`, i.e., `4`).
7. `rpact.summary.trim.zeroes`: if `TRUE` (default) zeroes will always displayed as "0", e.g. "0.000" will become "0".

Example: `options("rpact.summary.intervalFormat" = "%s - %s")`

How to get help for generic functions

Click on the link of a generic in the list above to go directly to the help documentation of the `rpact` specific implementation of the generic. Note that you can use the R function `methods` to get all the methods of a generic and to identify the object specific name of it, e.g., use `methods("plot")` to get all the methods for the `plot` generic. There you can find, e.g., `plot.AnalysisResults` and obtain the specific help documentation linked above by typing `?plot.AnalysisResults`.

`SummaryFactory` *Summary Factory*

Description

Basic class for summaries

`testPackage` *Test and Validate the rpact Package Installation*

Description

This function ensures the correct installation of the `rpact` package by performing various tests. It supports a comprehensive validation process, essential for GxP compliance and other regulatory requirements.

Usage

```
testPackage(
  outDir = ".",
  ...,
  completeUnitTestSetEnabled = TRUE,
  connection = list(token = NULL, secret = NULL),
  testFileDirectory = NA_character_,
  downloadTestsOnly = FALSE,
  addWarningDetailsToReport = TRUE,
  reportType = c("compact", "detailed", "Rout"),
  testInstalledBasicPackages = TRUE,
  scope = c("basic", "devel", "both", "internet", "all"),
  openHtmlReport = TRUE,
  keepSourceFiles = FALSE,
  reportFileBaseName = "rpact_test_result_report",
  metaData = list(),
  metaDataFile = NULL
)
```

Arguments

`outDir`	The absolute path to the output directory where all test results will be saved. By default, the current working directory is used.
`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`completeUnitTestSetEnabled`	If `TRUE` (default), all existing unit tests will be executed; if `FALSE`, only a subset of tests is run.
`connection`	A `list` allowing owners of the rpact validation documentation to provide `token` and `secret` credentials for full access to unit tests, enabling them to meet regulatory requirements (see www.rpact.com for more information).
`testFileDirectory`	An optional path pointing to a local directory containing test files.
`downloadTestsOnly`	If `TRUE`, the unit test files are only downloaded and not executed. Default is `FALSE`.
`addWarningDetailsToReport`	If `TRUE`, additional warning details are included in the test report. Default is `TRUE`.
`reportType`	The type of report to generate. Can be `"compact"`, `"detailed"`, or `"Rout"`.
`testInstalledBasicPackages`	If `TRUE`, tests for installed basic R packages are included, default is `TRUE`. For more information, see `testInstalledBasic`.
`scope`	The scope of the basic R package tests to run. Can be `"basic"`, `"devel"`, `"both"`, `"internet"`, or `"all"`. Default is `"basic"`. For more information, see `testInstalledBasic`. Only available if `testInstalledBasicPackages = TRUE`.
`openHtmlReport`	If `TRUE`, the HTML report is opened after the tests are completed, default is `TRUE`.

keepSourceFiles
: If TRUE, the source files are kept after the tests are completed. A copy of them can be found in the subdirectory src.

reportFileBaseName
: The base name for the report files (without path or extension).

metaData
: A named list containing additional metadata to be included in the test report. This can include information such as company, container name, host name, etc. Default is an empty list. For more information, see writeKeyValueFile and readKeyValueFile.

metaDataFile
: An optional path to a text file containing metadata: one entry per line in the form KEY=VALUE. For more information, see writeKeyValueFile and readKeyValueFile.

Details

This function is integral to the installation qualification (IQ) process of the rpact package, ensuring it meets quality standards and functions as expected. A directory named rpact-tests is created within the specified output directory, where all test files are downloaded from a secure resource and executed. Results are saved in the file testthat.Rout, located in the rpact-tests directory.

Installation qualification is a critical step in the validation process. Without successful IQ, the package cannot be considered fully validated. To gain access to the full set of unit tests, users must provide token and secret credentials, which are distributed to members of the rpact user group as part of the validation documentation. For more information, see vignette rpact_installation_qualification.

Value

Invisibly returns an InstallationQualificationResult) object.

References

For more information, please visit: https://www.rpact.org/vignettes/utilities/rpact_installation_qualification/

Examples

```
## Not run:
# Set the output directory
setwd("/path/to/output")

# Basic usage
testPackage()

# Perform all unit tests with access credentials
testPackage(
    connection = list(
        token = "your_token_here",
        secret = "your_secret_here"
    )
)

# Download test files without executing them
testPackage(downloadTestsOnly = TRUE)

## End(Not run)
```

test_plan_section *Test Plan Section*

Description

The section title or description will be used in the formal validation documentation. For more information visit https://www.rpact.com

Usage

```
test_plan_section(section)
```

Arguments

section	The section title or description.

TrialDesign *Basic Trial Design*

Description

Basic class for trial designs.

Details

TrialDesign is the basic class for

- TrialDesignFisher,
- TrialDesignGroupSequential,
- TrialDesignInverseNormal, and
- TrialDesignConditionalDunnett.

Fields

kMax The maximum number of stages K. Is a numeric vector of length 1 containing a whole number.

alpha The significance level alpha, default is 0.025. Is a numeric vector of length 1 containing a value between 0 and 1.

stages The stage numbers of the trial. Is a numeric vector of length kMax containing whole numbers.

informationRates The information rates (that must be fixed prior to the trial), default is (1:kMax) / kMax. Is a numeric vector of length kMax containing values between 0 and 1.

userAlphaSpending The user defined alpha spending. Contains the cumulative alpha-spending (type I error rate) up to each interim stage. Is a numeric vector of length kMax containing values between 0 and 1.

criticalValues The critical values for each stage of the trial. Is a numeric vector of length kMax.

stageLevels The adjusted significance levels to reach significance in a group sequential design. Is a numeric vector of length kMax containing values between 0 and 1.

`alphaSpent` The cumulative alpha spent at each stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`bindingFutility` If `TRUE`, the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`) Is a logical vector of length 1.

`tolerance` The numerical tolerance, default is `1e-06`. Is a numeric vector of length 1.

`TrialDesignCharacteristics`
Trial Design Characteristics

Description

Class for trial design characteristics.

Details

`TrialDesignCharacteristics` contains all fields required to collect the characteristics of a design. This object should not be created directly; use `getDesignCharacteristics` with suitable arguments to create it.

Fields

`nFixed` The sample size in a fixed (one-stage) design. Is a positive numeric vector.

`shift` The shift value for group sequential test characteristics. Is a numeric vector of length 1.

`inflationFactor` The relative increase of maximum sample size in a group sequential design as compared to the fixed sample size case. Is a positive numeric vector of length 1.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`information` The information over stages needed to achieve power of the specified design. Is a numeric vector of length `kMax`.

`power` The one-sided power at each stage of the trial. Is a numeric vector of length `kMax` containing values between 0 and 1.

`rejectionProbabilities` The rejection probabilities over treatments arms or populations and stages. Is a numeric vector.

`futilityProbabilities` The overall probabilities of stopping the trial for futility. Is a numeric vector of length `kMax` minus 1 containing values between 0 and 1.

`averageSampleNumber1` The expected sample size under H1. Is a positive numeric vector of length 1.

`averageSampleNumber01` The expected sample size for a value between H0 and H1. Is a positive numeric vector of length 1.

`averageSampleNumber0` The expected sample size under H0. Is a positive numeric vector of length 1.

See Also

`getDesignCharacteristics` for getting the design characteristics.

`TrialDesignConditionalDunnett`
Conditional Dunnett Design

Description

Trial design for conditional Dunnett tests.

Details

This object should not be created directly; use `getDesignConditionalDunnett` with suitable arguments to create a conditional Dunnett test design.

Fields

`kMax` The maximum number of stages `K`. Is a numeric vector of length 1 containing a whole number.

`alpha` The significance level alpha, default is 0.025. Is a numeric vector of length 1 containing a value between 0 and 1.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`informationRates` The information rates (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. Is a numeric vector of length `kMax` containing values between 0 and 1.

`userAlphaSpending` The user defined alpha spending. Contains the cumulative alpha-spending (type I error rate) up to each interim stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`criticalValues` The critical values for each stage of the trial. Is a numeric vector of length `kMax`.

`stageLevels` The adjusted significance levels to reach significance in a group sequential design. Is a numeric vector of length `kMax` containing values between 0 and 1.

`alphaSpent` The cumulative alpha spent at each stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`bindingFutility` If `TRUE`, the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`) Is a logical vector of length 1.

`tolerance` The numerical tolerance, default is `1e-06`. Is a numeric vector of length 1.

`informationAtInterim` The information to be expected at interim, default is informationAtInterim = 0.5. Is a numeric vector of length 1 containing a value between 0 and 1.

`secondStageConditioning` The way the second stage p-values are calculated within the closed system of hypotheses. If `FALSE`, the unconditional adjusted p-values are used, otherwise conditional adjusted p-values are calculated. Is a logical vector of length 1.

`sided` Describes if the alternative is one-sided (`1`) or two-sided (`2`). Is a numeric vector of length 1 containing a whole number.

See Also

`getDesignConditionalDunnett` for creating a conditional Dunnett test design.

`TrialDesignFisher` *Fisher Design*

Description

Trial design for Fisher's combination test.

Details

This object should not be created directly; use `getDesignFisher` with suitable arguments to create a Fisher combination test design.

Fields

`kMax` The maximum number of stages `K`. Is a numeric vector of length 1 containing a whole number.

`alpha` The significance level alpha, default is 0.025. Is a numeric vector of length 1 containing a value between 0 and 1.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`informationRates` The information rates (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. Is a numeric vector of length `kMax` containing values between 0 and 1.

`userAlphaSpending` The user defined alpha spending. Contains the cumulative alpha-spending (type I error rate) up to each interim stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`criticalValues` The critical values for each stage of the trial. Is a numeric vector of length `kMax`.

`stageLevels` The adjusted significance levels to reach significance in a group sequential design. Is a numeric vector of length `kMax` containing values between 0 and 1.

`alphaSpent` The cumulative alpha spent at each stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`bindingFutility` If `TRUE`, the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`) Is a logical vector of length 1.

`tolerance` The numerical tolerance, default is `1e-06`. Is a numeric vector of length 1.

`method` "equalAlpha", "fullAlpha", "noInteraction", or "userDefinedAlpha", default is "equalAlpha". For details, see Wassmer, 1999, doi: 10.1002/(SICI)1521-4036(199906)41:3%3C279::AID-BIMJ279%3E3.0.CO;2-V.

`alpha0Vec` The stopping for futility bounds for stage-wise p-values in Fisher's combination test. Is a numeric vector of length `kMax` minus 1 containing values between 0 and 1.

`scale` The scale for Fisher's combination test. Numeric vector of length `kMax-1` that applies to Fisher's design with unequally spaced information rates. Is a numeric vector of length `kMax` minus 1 containing values between 0 and 1.

`nonStochasticCurtailment` If `TRUE`, the stopping rule is based on the phenomenon of non-stochastic curtailment rather than stochastic reasoning. Is a logical vector of length 1.

`sided` Describes if the alternative is one-sided (`1`) or two-sided (`2`). Is a numeric vector of length 1 containing a whole number.

`simAlpha` The observed alpha error if simulations have been performed. Is a numeric vector of length 1 containing a value between 0 and 1.

`iterations` The number of iterations used for simulations. Is a numeric vector of length 1 containing a whole number.

`seed` The seed used for random number generation. Is a numeric vector of length 1.

See Also

`getDesignFisher` for creating a Fisher combination test design.

TrialDesignGroupSequential
Group Sequential Design

Description

Trial design for group sequential design.

Details

This object should not be created directly; use `getDesignGroupSequential()` with suitable arguments to create a group sequential design.

Fields

`kMax` The maximum number of stages `K`. Is a numeric vector of length 1 containing a whole number.

`alpha` The significance level alpha, default is 0.025. Is a numeric vector of length 1 containing a value between 0 and 1.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`informationRates` The information rates (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. Is a numeric vector of length `kMax` containing values between 0 and 1.

`userAlphaSpending` The user defined alpha spending. Contains the cumulative alpha-spending (type I error rate) up to each interim stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`criticalValues` The critical values for each stage of the trial. Is a numeric vector of length `kMax`.

`stageLevels` The adjusted significance levels to reach significance in a group sequential design. Is a numeric vector of length `kMax` containing values between 0 and 1.

`alphaSpent` The cumulative alpha spent at each stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`bindingFutility` If `TRUE`, the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`) Is a logical vector of length 1.

`tolerance` The numerical tolerance, default is `1e-06`. Is a numeric vector of length 1.

`typeOfDesign` The type of design. Is a character vector of length 1.

`beta` The Type II error rate necessary for providing sample size calculations (e.g., in `getSampleSizeMeans`), beta spending function designs, or optimum designs, default is `0.20`. Is a numeric vector of length 1 containing a value between 0 and 1.

`deltaWT` Delta for Wang & Tsiatis Delta class. Is a numeric vector of length 1.

deltaPT1 Delta1 for Pampallona & Tsiatis class rejecting H0 boundaries. Is a numeric vector of length 1.

deltaPT0 Delta0 for Pampallona & Tsiatis class rejecting H1 (accepting H0) boundaries. Is a numeric vector of length 1.

futilityBounds The futility bounds for each stage of the trial. Is a numeric vector of length kMax.

gammaA The parameter for the alpha spending function. Is a numeric vector of length 1.

gammaB The parameter for the beta spending function. Is a numeric vector of length 1.

optimizationCriterion The optimization criterion for optimum design within the Wang & Tsiatis class ("ASNH1", "ASNIFH1", "ASNsum"), default is "ASNH1".

sided Describes if the alternative is one-sided (1) or two-sided (2). Is a numeric vector of length 1 containing a whole number.

betaSpent The cumulative beta level spent at each stage of the trial. Only applicable for beta-spending designs. Is a numeric vector of length kMax containing values between 0 and 1.

typeBetaSpending The type of beta spending. Is a character vector of length 1.

userBetaSpending The user defined beta spending. Contains the cumulative beta-spending up to each interim stage. Is a numeric vector of length kMax containing values between 0 and 1.

efficacyStops Logical vector indicating efficacy stops Is a logical vector of length kMax minus 1.

futilityStops Logical vector indicating futility stops Is a logical vector of length kMax minus 1.

power The one-sided power at each stage of the trial. Is a numeric vector of length kMax containing values between 0 and 1.

twoSidedPower Specifies if power is defined two-sided at each stage of the trial. Is a logical vector of length 1.

constantBoundsHP The constant bounds up to stage kMax - 1 for the Haybittle & Peto design (default is 3). Is a numeric vector of length 1.

betaAdjustment If TRUE, beta spending values are linearly adjusted if an overlapping of decision regions for futility stopping at earlier stages occurs. Only applicable for two-sided beta-spending designs. Is a logical vector of length 1.

delayedInformation Delay of information for delayed response designs. Is a numeric vector of length kMax minus 1 containing values between 0 and 1.

decisionCriticalValues The decision critical values for each stage of the trial in a delayed response design. Is a numeric vector of length kMax.

reversalProbabilities The probability to switch from stopping the trial for success (or futility) and reaching non-rejection (or rejection) in a delayed response design. Is a numeric vector of length kMax minus 1 containing values between 0 and 1.

See Also

getDesignGroupSequential() for creating a group sequential design.

`TrialDesignInverseNormal`

Inverse Normal Design

Description

Trial design for inverse normal method.

Details

This object should not be created directly; use `getDesignInverseNormal()` with suitable arguments to create a inverse normal design.

Fields

`kMax` The maximum number of stages `K`. Is a numeric vector of length 1 containing a whole number.

`alpha` The significance level alpha, default is 0.025. Is a numeric vector of length 1 containing a value between 0 and 1.

`stages` The stage numbers of the trial. Is a numeric vector of length `kMax` containing whole numbers.

`informationRates` The information rates (that must be fixed prior to the trial), default is `(1:kMax) / kMax`. Is a numeric vector of length `kMax` containing values between 0 and 1.

`userAlphaSpending` The user defined alpha spending. Contains the cumulative alpha-spending (type I error rate) up to each interim stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`criticalValues` The critical values for each stage of the trial. Is a numeric vector of length `kMax`.

`stageLevels` The adjusted significance levels to reach significance in a group sequential design. Is a numeric vector of length `kMax` containing values between 0 and 1.

`alphaSpent` The cumulative alpha spent at each stage. Is a numeric vector of length `kMax` containing values between 0 and 1.

`bindingFutility` If `TRUE`, the calculation of the critical values is affected by the futility bounds and the futility threshold is binding in the sense that the study must be stopped if the futility condition was reached (default is `FALSE`) Is a logical vector of length 1.

`tolerance` The numerical tolerance, default is `1e-06`. Is a numeric vector of length 1.

`typeOfDesign` The type of design. Is a character vector of length 1.

`beta` The Type II error rate necessary for providing sample size calculations (e.g., in `getSampleSizeMeans`), beta spending function designs, or optimum designs, default is `0.20`. Is a numeric vector of length 1 containing a value between 0 and 1.

`deltaWT` Delta for Wang & Tsiatis Delta class. Is a numeric vector of length 1.

`deltaPT1` Delta1 for Pampallona & Tsiatis class rejecting H0 boundaries. Is a numeric vector of length 1.

`deltaPT0` Delta0 for Pampallona & Tsiatis class rejecting H1 (accepting H0) boundaries. Is a numeric vector of length 1.

`futilityBounds` The futility bounds for each stage of the trial. Is a numeric vector of length `kMax`.

`gammaA` The parameter for the alpha spending function. Is a numeric vector of length 1.

`gammaB` The parameter for the beta spending function. Is a numeric vector of length 1.

optimizationCriterion The optimization criterion for optimum design within the Wang & Tsiatis class ("ASNH1", "ASNIFH1", "ASNsum"), default is "ASNH1".

sided Describes if the alternative is one-sided (1) or two-sided (2). Is a numeric vector of length 1 containing a whole number.

betaSpent The cumulative beta level spent at each stage of the trial. Only applicable for beta-spending designs. Is a numeric vector of length kMax containing values between 0 and 1.

typeBetaSpending The type of beta spending. Is a character vector of length 1.

userBetaSpending The user defined beta spending. Contains the cumulative beta-spending up to each interim stage. Is a numeric vector of length kMax containing values between 0 and 1.

efficacyStops Logical vector indicating efficacy stops Is a logical vector of length kMax minus 1.

futilityStops Logical vector indicating futility stops Is a logical vector of length kMax minus 1.

power The one-sided power at each stage of the trial. Is a numeric vector of length kMax containing values between 0 and 1.

twoSidedPower Specifies if power is defined two-sided at each stage of the trial. Is a logical vector of length 1.

constantBoundsHP The constant bounds up to stage kMax - 1 for the Haybittle & Peto design (default is 3). Is a numeric vector of length 1.

betaAdjustment If TRUE, beta spending values are linearly adjusted if an overlapping of decision regions for futility stopping at earlier stages occurs. Only applicable for two-sided beta-spending designs. Is a logical vector of length 1.

delayedInformation Delay of information for delayed response designs. Is a numeric vector of length kMax minus 1 containing values between 0 and 1.

decisionCriticalValues The decision critical values for each stage of the trial in a delayed response design. Is a numeric vector of length kMax.

reversalProbabilities The probability to switch from stopping the trial for success (or futility) and reaching non-rejection (or rejection) in a delayed response design. Is a numeric vector of length kMax minus 1 containing values between 0 and 1.

See Also

getDesignInverseNormal() for creating a inverse normal design.

TrialDesignPlan *Basic Trial Design Plan*

Description

Basic class for trial design plans.

Details

TrialDesignPlan is the basic class for

- TrialDesignPlanMeans,
- TrialDesignPlanRates, and
- TrialDesignPlanSurvival.

`TrialDesignPlanCountData`

Trial Design Plan Count Data

Description

Trial design plan for count data.

Details

This object cannot be created directly; use `getSampleSizeCounts()` with suitable arguments to create a design plan for a dataset of rates.

Fields

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`groups` The group numbers. Is a numeric vector.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`optimumAllocationRatio` The allocation ratio that is optimum with respect to the overall sample size at given power. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`lambda1` The assumed hazard rate in the treatment group. Is a numeric vector of length `kMax`.

`lambda2` The assumed hazard rate in the reference group. Is a numeric vector of length 1.

`lambda` A numeric value or vector that represents the assumed rate of a homogeneous Poisson process in the pooled treatment groups Is a numeric vector.

`theta` A vector of standardized effect sizes (theta values). Is a numeric vector.

`nFixed` The sample size in a fixed (one-stage) design. Is a positive numeric vector.

`nFixed1` The sample size in treatment arm 1 in a fixed (one-stage) design. Is a positive numeric vector.

`nFixed2` The sample size in treatment arm 2 in a fixed (one-stage) design. Is a positive numeric vector.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`maxNumberOfSubjects1` The maximum number of subjects in treatment arm 1. Is a numeric vector.

`maxNumberOfSubjects2` The maximum number of subjects in treatment arm 2. Is a numeric vector.

`overallReject` The overall rejection probability. Is a numeric vector.

`rejectPerStage` The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

`futilityStop` In simulation results data set: indicates whether trial is stopped for futility or not.

`futilityPerStage` The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

`earlyStop` The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

`overdispersion` A numeric value that represents the assumed overdispersion of the negative binomial distribution Is a numeric vector.

`fixedExposureTime` If specified, the fixed time of exposure per subject for count data Is a numeric vector.

`accrualTime` The assumed accrual time intervals for the study. Is a numeric vector.

`accrualIntensity` The absolute accrual intensities. Is a numeric vector of length `kMax`.

`followUpTime` The assumed follow-up time for the study. Is a numeric vector of length 1.

`calendarTime` The calendar time Is a numeric vector.

`expectedStudyDurationH1` The expected study duration under H1 Is a numeric vector.

`studyTime` The study time Is a numeric vector.

`numberOfSubjects` In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

`expectedNumberOfSubjectsH1` The expected number of subjects under H1. Is a numeric vector.

`informationOverStages` The information over stages Is a numeric vector.

`expectedInformationH0` The expected information under H0 Is a numeric vector.

`expectedInformationH01` The expected information under H0/H1 Is a numeric vector.

`expectedInformationH1` The expected information under H1 Is a numeric vector.

`maxInformation` The maximum information. Is a numeric vector of length 1 containing a whole number.

`futilityBoundsPValueScale` The futility bounds for each stage of the trial on the p-value scale. Is a numeric matrix.

`TrialDesignPlanMeans` *Trial Design Plan Means*

Description

Trial design plan for means.

Details

This object cannot be created directly; use `getSampleSizeMeans()` with suitable arguments to create a design plan for a dataset of means.

Fields

`meanRatio` Specifies if the sample size for one-sided testing of H0: `mu1/mu2 = thetaH0` has been calculated. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`alternative` The alternative hypothesis value(s) for testing means. Is a numeric vector.

`stDev` The standard deviation used for sample size and power calculation. Is a numeric vector of length 1.

groups The group numbers. Is a numeric vector.

allocationRatioPlanned The planned allocation ratio (n1 / n2) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

optimumAllocationRatio The allocation ratio that is optimum with respect to the overall sample size at given power. Is a logical vector of length 1.

directionUpper Specifies the direction of the alternative, only applicable for one-sided testing. Default is TRUE which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

effect The effect for randomly creating normally distributed responses. Is a numeric vector of length kMax.

overallReject The overall rejection probability. Is a numeric vector.

rejectPerStage The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

expectedNumberOfSubjects The expected number of subjects under specified alternative.

nFixed The sample size in a fixed (one-stage) design. Is a positive numeric vector.

nFixed1 The sample size in treatment arm 1 in a fixed (one-stage) design. Is a positive numeric vector.

nFixed2 The sample size in treatment arm 2 in a fixed (one-stage) design. Is a positive numeric vector.

informationRates The information rates (that must be fixed prior to the trial), default is (1:kMax) / kMax. Is a numeric vector of length kMax containing values between 0 and 1.

maxNumberOfSubjects The maximum number of subjects for power calculations. Is a numeric vector.

maxNumberOfSubjects1 The maximum number of subjects in treatment arm 1. Is a numeric vector.

maxNumberOfSubjects2 The maximum number of subjects in treatment arm 2. Is a numeric vector.

numberOfSubjects In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

numberOfSubjects1 In simulation results data set: The number of subjects under consideration in treatment arm 1 when the interim analysis takes place.

numberOfSubjects2 In simulation results data set: The number of subjects under consideration in treatment arm 2 when the interim analysis takes place.

expectedNumberOfSubjectsH0 The expected number of subjects under H0. Is a numeric vector.

expectedNumberOfSubjectsH01 The expected number of subjects under a value between H0 and H1. Is a numeric vector.

expectedNumberOfSubjectsH1 The expected number of subjects under H1. Is a numeric vector.

criticalValuesEffectScale The critical values for each stage of the trial on the effect size scale.

criticalValuesEffectScaleLower The lower critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

`criticalValuesEffectScaleUpper` The upper critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

`criticalValuesPValueScale` The critical values for each stage of the trial on the p-value scale.

`futilityBoundsEffectScale` The futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsEffectScaleLower` The lower futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsEffectScaleUpper` The upper futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsPValueScale` The futility bounds for each stage of the trial on the p-value scale. Is a numeric matrix.

`TrialDesignPlanRates` *Trial Design Plan Rates*

Description

Trial design plan for rates.

Details

This object cannot be created directly; use `getSampleSizeRates()` with suitable arguments to create a design plan for a dataset of rates.

Fields

`riskRatio` Specifies if the sample size for one-sided testing of H0: `pi1 / pi2 = thetaH0` has been calculated. Is a logical vector of length 1.

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`normalApproximation` Describes if a normal approximation was used when calculating p-values. Default for means is `FALSE` and `TRUE` for rates and hazard ratio. Is a logical vector of length 1.

`conservative` Conservative sample size calculation enabled or not Is a logical vector of length 1.

`pi1` The assumed probability or probabilities in the active treatment group in two-group designs, or the alternative probability for a one-group design.

`pi2` The assumed probability in the reference group for two-group designs. Is a numeric vector of length 1 containing a value between 0 and 1.

`groups` The group numbers. Is a numeric vector.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`optimumAllocationRatio` The allocation ratio that is optimum with respect to the overall sample size at given power. Is a logical vector of length 1.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`effect` The effect for randomly creating normally distributed responses. Is a numeric vector of length `kMax`.

overallReject The overall rejection probability. Is a numeric vector.

rejectPerStage The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

expectedNumberOfSubjects The expected number of subjects under specified alternative.

nFixed The sample size in a fixed (one-stage) design. Is a positive numeric vector.

nFixed1 The sample size in treatment arm 1 in a fixed (one-stage) design. Is a positive numeric vector.

nFixed2 The sample size in treatment arm 2 in a fixed (one-stage) design. Is a positive numeric vector.

informationRates The information rates (that must be fixed prior to the trial), default is (1:kMax) / kMax. Is a numeric vector of length kMax containing values between 0 and 1.

maxNumberOfSubjects The maximum number of subjects for power calculations. Is a numeric vector.

maxNumberOfSubjects1 The maximum number of subjects in treatment arm 1. Is a numeric vector.

maxNumberOfSubjects2 The maximum number of subjects in treatment arm 2. Is a numeric vector.

numberOfSubjects In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

numberOfSubjects1 In simulation results data set: The number of subjects under consideration in treatment arm 1 when the interim analysis takes place.

numberOfSubjects2 In simulation results data set: The number of subjects under consideration in treatment arm 2 when the interim analysis takes place.

expectedNumberOfSubjectsH0 The expected number of subjects under H0. Is a numeric vector.

expectedNumberOfSubjectsH01 The expected number of subjects under a value between H0 and H1. Is a numeric vector.

expectedNumberOfSubjectsH1 The expected number of subjects under H1. Is a numeric vector.

criticalValuesEffectScale The critical values for each stage of the trial on the effect size scale.

criticalValuesEffectScaleLower The lower critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

criticalValuesEffectScaleUpper The upper critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

criticalValuesPValueScale The critical values for each stage of the trial on the p-value scale.

futilityBoundsEffectScale The futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

futilityBoundsEffectScaleLower The lower futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

futilityBoundsEffectScaleUpper The upper futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

futilityBoundsPValueScale The futility bounds for each stage of the trial on the p-value scale. Is a numeric matrix.

`TrialDesignPlanSurvival` *Trial Design Plan Survival*

Description

Trial design plan for survival data.

Details

This object cannot be created directly; use `getSampleSizeSurvival()` with suitable arguments to create a design plan for a dataset of survival data.

Fields

`thetaH0` The difference or assumed effect under H0. Is a numeric vector of length 1.

`typeOfComputation` The type of computation used, either `"Schoenfeld"`, `"Freedman"`, or `"HsiehFreedman"`.

`directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing. Default is `TRUE` which means that larger values of the test statistics yield smaller p-values. Is a logical vector of length 1.

`pi1` The assumed event rate in the treatment group. Is a numeric vector of length `kMax` containing values between 0 and 1.

`pi2` The assumed event rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

`median1` The assumed median survival time in the treatment group. Is a numeric vector.

`median2` The assumed median survival time in the reference group. Is a numeric vector of length 1.

`lambda1` The assumed hazard rate in the treatment group. Is a numeric vector of length `kMax`.

`lambda2` The assumed hazard rate in the reference group. Is a numeric vector of length 1.

`hazardRatio` The hazard ratios under consideration. Is a numeric vector of length `kMax`.

`maxNumberOfSubjects` The maximum number of subjects for power calculations. Is a numeric vector.

`maxNumberOfSubjects1` The maximum number of subjects in treatment arm 1. Is a numeric vector.

`maxNumberOfSubjects2` The maximum number of subjects in treatment arm 2. Is a numeric vector.

`maxNumberOfEvents` The maximum number of events for power calculations. Is a positive numeric vector of length `kMax`.

`allocationRatioPlanned` The planned allocation ratio (`n1 / n2`) for the groups. For multi-arm designs, it is the allocation ratio relating the active arm(s) to the control. Is a positive numeric vector of length 1.

`optimumAllocationRatio` The allocation ratio that is optimum with respect to the overall sample size at given power. Is a logical vector of length 1.

`accountForObservationTimes` If `FALSE`, only the event rates are used for the calculation of the maximum number of subjects. Is a logical vector of length 1.

eventTime The assumed time under which the event rates are calculated. Is a numeric vector of length 1.

accrualTime The assumed accrual time intervals for the study. Is a numeric vector.

totalAccrualTime The total accrual time, i.e., the maximum of accrualTime. Is a positive numeric vector of length 1.

accrualIntensity The absolute accrual intensities. Is a numeric vector of length kMax.

accrualIntensityRelative The relative accrual intensities.

kappa The shape of the Weibull distribution if kappa!=1. Is a numeric vector of length 1.

piecewiseSurvivalTime The time intervals for the piecewise definition of the exponential survival time cumulative distribution function. Is a numeric vector.

followUpTime The assumed follow-up time for the study. Is a numeric vector of length 1.

dropoutRate1 The assumed drop-out rate in the treatment group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutRate2 The assumed drop-out rate in the control group. Is a numeric vector of length 1 containing a value between 0 and 1.

dropoutTime The assumed time for drop-out rates in the control and treatment group. Is a numeric vector of length 1.

chi The calculated event probability at end of trial. Is a numeric vector.

expectedNumberOfEvents The expected number of events under specified alternative. Is a numeric vector.

eventsFixed The number of events in a fixed sample size design. Is a numeric vector.

nFixed The sample size in a fixed (one-stage) design. Is a positive numeric vector.

nFixed1 The sample size in treatment arm 1 in a fixed (one-stage) design. Is a positive numeric vector.

nFixed2 The sample size in treatment arm 2 in a fixed (one-stage) design. Is a positive numeric vector.

overallReject The overall rejection probability. Is a numeric vector.

rejectPerStage The probability to reject a hypothesis per stage of the trial. Is a numeric matrix.

futilityStop In simulation results data set: indicates whether trial is stopped for futility or not.

futilityPerStage The per-stage probabilities of stopping the trial for futility. Is a numeric matrix.

earlyStop The probability to stopping the trial either for efficacy or futility. Is a numeric vector.

informationRates The information rates (that must be fixed prior to the trial), default is (1:kMax) / kMax. Is a numeric vector of length kMax containing values between 0 and 1.

analysisTime The estimated time of analysis. Is a numeric matrix.

studyDurationH1 The study duration under the alternative hypothesis. Is a positive numeric vector.

studyDuration The study duration for specified effect size. Is a positive numeric vector.

maxStudyDuration The maximum study duration in survival designs. Is a numeric vector.

eventsPerStage Deprecated: use singleEventsPerStage or cumulativeEventsPerStage instead Is a numeric matrix.

singleEventsPerStage The single number of events per stage. Is a numeric matrix.

cumulativeEventsPerStage The cumulative number of events per stage. Is a numeric matrix.

`expectedEventsH0` The expected number of events under H0. Is a numeric vector.

`expectedEventsH01` The expected number of events under a value between H0 and H1. Is a numeric vector.

`expectedEventsH1` The expected number of events under H1. Is a numeric vector.

`numberOfSubjects` In simulation results data set: The number of subjects under consideration when the interim analysis takes place.

`numberOfSubjects1` In simulation results data set: The number of subjects under consideration in treatment arm 1 when the interim analysis takes place.

`numberOfSubjects2` In simulation results data set: The number of subjects under consideration in treatment arm 2 when the interim analysis takes place.

`expectedNumberOfSubjectsH1` The expected number of subjects under H1. Is a numeric vector.

`expectedNumberOfSubjects` The expected number of subjects under specified alternative.

`criticalValuesEffectScale` The critical values for each stage of the trial on the effect size scale.

`criticalValuesEffectScaleLower` The lower critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

`criticalValuesEffectScaleUpper` The upper critical values for each stage of the trial on the effect size scale. Is a numeric matrix.

`criticalValuesPValueScale` The critical values for each stage of the trial on the p-value scale.

`futilityBoundsEffectScale` The futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsEffectScaleLower` The lower futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsEffectScaleUpper` The upper futility bounds for each stage of the trial on the effect size scale. Is a numeric matrix.

`futilityBoundsPValueScale` The futility bounds for each stage of the trial on the p-value scale. Is a numeric matrix.

`TrialDesignSet` *Class for trial design sets.*

Description

`TrialDesignSet` is a class for creating a collection of different trial designs.

Details

This object cannot be created directly; better use `getDesignSet()` with suitable arguments to create a set of designs.

Fields

`designs` The trial designs to be compared.

`design` The trial design.

`variedParameters` A character vector containing the names of the parameters that vary between designs.

See Also

`getDesignSet()`

utilitiesForPiecewiseExponentialDistribution
The Piecewise Exponential Distribution

Description

Distribution function, quantile function and random number generation for the piecewise exponential distribution.

Usage

```
getPiecewiseExponentialDistribution(
  time,
  ...,
  piecewiseSurvivalTime = NA_real_,
  piecewiseLambda = NA_real_,
  kappa = 1
)

ppwexp(t, ..., s = NA_real_, lambda = NA_real_, kappa = 1)

getPiecewiseExponentialQuantile(
  quantile,
  ...,
  piecewiseSurvivalTime = NA_real_,
  piecewiseLambda = NA_real_,
  kappa = 1
)

qpwexp(q, ..., s = NA_real_, lambda = NA_real_, kappa = 1)

getPiecewiseExponentialRandomNumbers(
  n,
  ...,
  piecewiseSurvivalTime = NA_real_,
  piecewiseLambda = NA_real_,
  kappa = 1
)

rpwexp(n, ..., s = NA_real_, lambda = NA_real_, kappa = 1)
```

Arguments

`...`	Ensures that all arguments (starting from the "...") are to be named and that a warning will be displayed if unknown arguments are passed.
`kappa`	A numeric value > 0. A `kappa != 1` will be used for the specification of the shape of the Weibull distribution. Default is `1`, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only `piecewiselambda` (as a single value) and `kappa` can be specified. This function is equivalent to `pweibull(t, shape = kappa, scale = 1 / lambda)` of

the stats package, i.e., the scale parameter is 1 / 'hazard rate'.
For example, getPiecewiseExponentialDistribution(time = 130, piecewiseLambda = 0.01, kappa = 4.2) and pweibull(q = 130, shape = 4.2, scale = 1 / 0.01) provide the same result.

t, time — Vector of time values.

s, piecewiseSurvivalTime — Vector of start times defining the "time pieces".

lambda, piecewiseLambda — Vector of lambda values (hazard rates) corresponding to the start times.

q, quantile — Vector of quantiles.

n — Number of observations.

Details

getPiecewiseExponentialDistribution() (short: ppwexp()), getPiecewiseExponentialQuantile() (short: qpwexp()), and getPiecewiseExponentialRandomNumbers() (short: rpwexp()) provide probabilities, quantiles, and random numbers according to a piecewise exponential or a Weibull distribution. The piecewise definition is performed through a vector of starting times (piecewiseSurvivalTime) and a vector of hazard rates (piecewiseLambda). You can also use a list that defines the starting times and piecewise lambdas together and define piecewiseSurvivalTime as this list. The list needs to have the form, e.g., piecewiseSurvivalTime <- list("0 - <6" = 0.025, "6 - <9" = 0.04, "9 - <15" = 0.015, ">=15" = 0.007) . For the Weibull case, you can also specify a shape parameter kappa in order to calculate probabilities, quantiles, or random numbers. In this case, no piecewise definition is possible, i.e., only piecewiseLambda (as a single value) and kappa need to be specified.

Value

A numeric value or vector will be returned.

Examples

```
## Not run:
# Calculate probabilties for a range of time values for a
# piecewise exponential distribution with hazard rates
# 0.025, 0.04, 0.015, and 0.007 in the intervals
# [0, 6), [6, 9), [9, 15), [15, Inf), respectively,
# and re-return the time values:
piecewiseSurvivalTime <- list(
    "0 - <6"   = 0.025,
    "6 - <9"   = 0.04,
    "9 - <15"  = 0.015,
    ">=15"     = 0.01
)
y <- getPiecewiseExponentialDistribution(seq(0, 150, 15),
    piecewiseSurvivalTime = piecewiseSurvivalTime
)
getPiecewiseExponentialQuantile(y,
    piecewiseSurvivalTime = piecewiseSurvivalTime
)

## End(Not run)
```

utilitiesForSurvivalTrials *Survival Helper Functions for Conversion of Pi, Lambda, Median, and Hazard Ratio Values*

Description

Functions to convert pi, lambda, median, and hazard ratio values into each other.

Usage

```
getLambdaByPi(piValue, eventTime = 12, kappa = 1)

getLambdaByMedian(median, kappa = 1)

getHazardRatioByPi(pi1, pi2, eventTime = 12, kappa = 1)

getHazardRatioByLambda(lambda1, lambda2)

getHazardRatioByMedian(median1, median2, kappa = 1)

getLambda1ByLambda2AndHazardRatio(lambda2, hazardRatio)

getLambda2ByLambda1AndHazardRatio(lambda1, hazardRatio)

getPi1ByPi2AndHazardRatio(pi2, hazardRatio, eventTime = 12, kappa = 1)

getPi2ByPi1AndHazardRatio(pi1, hazardRatio, eventTime = 12, kappa = 1)

getPiByLambda(lambda, eventTime = 12, kappa = 1)

getPiByMedian(median, eventTime = 12, kappa = 1)

getMedianByLambda(lambda, kappa = 1)

getMedianByPi(piValue, eventTime = 12, kappa = 1)
```

Arguments

piValue, pi1, pi2, lambda, lambda1, lambda2, median, median1, median2	Value that shall be converted.
eventTime	The assumed time under which the event rates are calculated, default is 12.
kappa	A numeric value > 0. A kappa != 1 will be used for the specification of the shape of the Weibull distribution. Default is 1, i.e., the exponential survival distribution is used instead of the Weibull distribution. Note that the Weibull distribution cannot be used for the piecewise definition of the survival time distribution, i.e., only piecewiselambda (as a single value) and kappa can be specified. This function is equivalent to pweibull(t, shape = kappa, scale = 1 / lambda) of the stats package, i.e., the scale parameter is 1 / 'hazard rate'. For example, getPiecewiseExponentialDistribution(time = 130, piecewiseLambda

= 0.01, kappa = 4.2) and pweibull(q = 130, shape = 4.2, scale = 1 / 0.01) provide the same result.

Details

Can be used, e.g., to convert pi, median, or lambda values into pi, median, lambda, or hazard ratio values, e.g, for usage in getSampleSizeSurvival() or getPowerSurvival().

Value

Returns a numeric value or vector will be returned.

---

writeDataset *Write Dataset*

Description

Writes a dataset to a CSV file.

Usage

```
writeDataset(
  dataset,
  file,
  ...,
  append = FALSE,
  quote = TRUE,
  sep = ",",
  eol = "\n",
  na = "NA",
  dec = ".",
  row.names = TRUE,
  col.names = NA,
  qmethod = "double",
  fileEncoding = "UTF-8"
)
```

Arguments

dataset	A dataset.
file	The target CSV file.
...	Further arguments to be passed to write.table.
append	Logical. Only relevant if file is a character string. If TRUE, the output is appended to the file. If FALSE, any existing file of the name is destroyed.
quote	The set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behavior on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.
sep	The field separator character. Values on each line of the file are separated by this character. If sep = "," (the default for writeDataset) the separator is a comma.

eol	The character(s) to print at the end of each line (row).
na	The string to use for missing values in the data.
dec	The character used in the file for decimal points.
row.names	Either a logical value indicating whether the row names of dataset are to be written along with dataset, or a character vector of row names to be written.
col.names	Either a logical value indicating whether the column names of dataset are to be written along with dataset, or a character vector of column names to be written. See the section on 'CSV files' for the meaning of col.names = NA.
qmethod	A character string specifying how to deal with embedded double quote characters when quoting strings. Must be one of "double" (default in writeDataset) or "escape".
fileEncoding	Character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, the 'R Data Import/Export Manual' and 'Note'.

Details

writeDataset() is a wrapper function that coerces the dataset to a data frame and uses write.table to write it to a CSV file.

See Also

- writeDatasets() for writing multiple datasets,
- readDataset() for reading a single dataset,
- readDatasets() for reading multiple datasets.

Examples

```
## Not run:
datasetOfRates <- getDataset(
    n1 = c(11, 13, 12, 13),
    n2 = c(8, 10, 9, 11),
    events1 = c(10, 10, 12, 12),
    events2 = c(3, 5, 5, 6)
)
writeDataset(datasetOfRates, "dataset_rates.csv")

## End(Not run)
```

---

writeDatasets *Write Multiple Datasets*

Description

Writes a list of datasets to a CSV file.

Usage

```
writeDatasets(
  datasets,
  file,
  ...,
  append = FALSE,
  quote = TRUE,
  sep = ",",
  eol = "\n",
  na = "NA",
  dec = ".",
  row.names = TRUE,
  col.names = NA,
  qmethod = "double",
  fileEncoding = "UTF-8"
)
```

Arguments

datasets	A list of datasets.
file	The target CSV file.
...	Further arguments to be passed to write.table.
append	Logical. Only relevant if file is a character string. If TRUE, the output is appended to the file. If FALSE, any existing file of the name is destroyed.
quote	The set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behavior on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.
sep	The field separator character. Values on each line of the file are separated by this character. If sep = "," (the default for writeDatasets) the separator is a comma.
eol	The character(s) to print at the end of each line (row).
na	The string to use for missing values in the data.
dec	The character used in the file for decimal points.
row.names	Either a logical value indicating whether the row names of dataset are to be written along with dataset, or a character vector of row names to be written.
col.names	Either a logical value indicating whether the column names of dataset are to be written along with dataset, or a character vector of column names to be written. See the section on 'CSV files' for the meaning of col.names = NA.
qmethod	A character string specifying how to deal with embedded double quote characters when quoting strings. Must be one of "double" (default in writeDatasets) or "escape".
fileEncoding	Character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, the 'R Data Import/Export Manual' and 'Note'.

Details

The format of the CSV file is optimized for usage of readDatasets().

See Also

- writeDataset() for writing a single dataset,
- readDatasets() for reading multiple datasets,
- readDataset() for reading a single dataset.

Examples

```
## Not run:
d1 <- getDataset(
    n1 = c(11, 13, 12, 13),
    n2 = c(8, 10, 9, 11),
    events1 = c(10, 10, 12, 12),
    events2 = c(3, 5, 5, 6)
)
d2 <- getDataset(
    n1 = c(9, 13, 12, 13),
    n2 = c(6, 10, 9, 11),
    events1 = c(10, 10, 12, 12),
    events2 = c(4, 5, 5, 6)
)
datasets <- list(d1, d2)
writeDatasets(datasets, "datasets_rates.csv")

## End(Not run)
```

---

writeKeyValueFile *Write a key-value file (KEY=VALUE) from a named list*

Description

Writes a human-editable text file in a widely used key-value format (INI/.env-like): one entry per line in the form KEY=VALUE. Blank lines and comment lines (starting with # or ;) are allowed. Values are written as a single line; special characters are escaped so that the file remains one key per line.

Usage

```
writeKeyValueFile(
  keyValueList,
  filePath,
  ...,
  writeHeader = TRUE,
  sortKeys = FALSE,
  overwrite = TRUE,
  safeKeyCheck = TRUE
)
```

Arguments

keyValueList	A named list of scalar (length-1) atomic values. Supported value types: character, logical, integer, numeric. NA is supported and written as NA.
filePath	Path to the output file (e.g. "inst/tests/META.env").
...	Currently unused.
writeHeader	Logical; if TRUE, writes a short header comment.
sortKeys	Logical; if TRUE, keys are written in alphabetical order.
overwrite	Logical; if FALSE and the file exists, an error is raised.
safeKeyCheck	Logical; if TRUE, checks that keys only contain allowed characters. Set to FALSE to allow arbitrary keys, but be aware that this may lead to issues when reading the file back, as keys with special characters may not be parsed correctly.

Details

Keys are restricted to [A-Za-z0-9_.-] to keep the file portable and easy to edit.

UTF-8 handling: Values are normalized to UTF-8 on write. The file is written with UTF-8 encoding when supported by the platform. On Windows, UTF-8 is enforced via fileEncoding = "UTF-8". On other platforms, UTF-8 is typically the native encoding; we still normalize strings to UTF-8.

Value

Invisibly returns filePath.

Examples

```
## Not run:
keyValueList <- list(
    STUDY_NAME = "Trial A",
    MAX_PATIENTS = 150L,
    THRESHOLD = 0.075,
    NOTES = "First phase\nSecond phase"
)
filePath <- tempfile(fileext = ".txt")
writeKeyValueFile(
    keyValueList = keyValueList,
    filePath = filePath,
    writeHeader = TRUE,
    sortKeys = TRUE,
    overwrite = TRUE
)

## End(Not run)
```

Index