

Introduction to rpact

Gernot Wassmer and Friedrich Pahlke

BBS Training Course
Basel, 13 September 2022

What is rpact?

rpact / RPACT

- rpact

- Comprehensive validated R package, freely available on CRAN
- Design, simulation, and analysis of confirmatory adaptive group sequential designs
- Monograph by Wassmer and Brannath, Springer, 2016

→ www.rpact.org

- RPACT is a company which offers

- technical support for the rpact package
- consultancy and user training for clinical researchers using R
- enterprise R/Shiny software development services

→ www.rpact.com

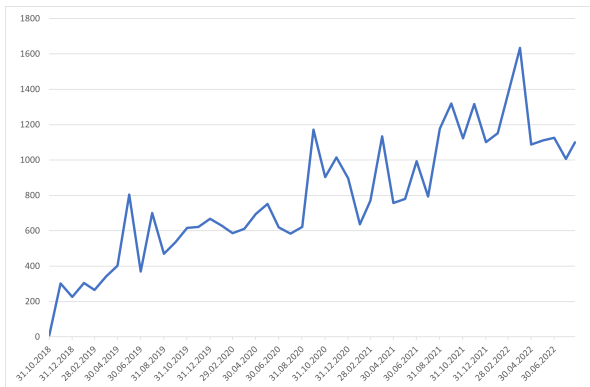


Company RPACT in Figures

- Founded in May 2017 by GW and FP
- Idea: open source development with help of “crowd funding”
- Currently supported by 21 companies
→ “Service Level Agreement” (SLA)
- 53 presentations and training courses since 2018

R package rpact in Figures

- 20 releases on CRAN since October 2018
- Comes with 25 vignettes
- CRAN download stats:



rpact – Functional Range

- Design
 - Comprehensive set of group sequential designs, e.g., Wang & Tsatis Δ -class, α -spending, β -spending, ...
 - Inverse normal design
 - Fisher's combination test
- Sample size and power calculation for
 - testing means (continuous endpoint)
 - testing rates (binary endpoint)
 - survival trials with, e.g.,
 - piecewise accrual time and intensity
 - flexible follow-up time specification
 - piecewise exponential survival time
 - fixed sample size design

rpact – Functional Range

- Analysis tool for
 - continuous, binary, and survival data
 - multi-arm adaptive trials
 - population enrichment designs
- Simulation tool for assessing adaptive strategies, e.g.,
 - sample size reassessment
 - treatment arm or population selection rules
 - different methodologies
- Graphical user interface:
Shiny app shiny.rpact.com

The rpact Package Concept

Package Concept – Workflow

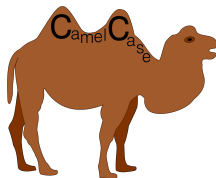
Usage inspired by the typical workflow in trial design and conduct:

- Everything is starting with a design, e.g.:
`design <- getDesignGroupSequential()`
- Find the optimal design parameters with help of rpact comparison tools: `getDesignSet()`
- Calculate the required sample size and power, e.g.:
`getSampleSizeMeans()` , `getPowerMeans()`
- Simulate specific characteristics of an adaptive design, e.g.:
`getSimulationMeans()`
- Collect your data, import it into R and create an rpact dataset:
`data <- getDataset()`
- Analyze your data:
`getAnalysisResults(design, data)`

Package Concept – Focus on Usability

Almost all functions, arguments, and objects are self-explanatory due to their names:

- `getDesign[GroupSequential/InverseNormal/Fisher]()`
- `getDesignCharacteristics()`
- `getSampleSize[Means/Rates/Survival]()`
- `getPower[Means/Rates/Survival]()`
- `getSimulation[MultiArm/Enrichment][Means/Rates/Survival]()`
- `getDataset()`
- `getAnalysisResults()`



Package Concept – Utilities

Several utility functions are available, e.g.:

- Survival helper functions:
 - `getAccrualTime()`
 - `getPiecewiseSurvivalTime()`
 - `getNumberOfSubjects()`
 - `getEventProbabilities()`
 - `getPiecewiseExponentialDistribution()`
- `getObjectRCode()`
- `testPackage()`: installation qualification on a client computer or company server (→ unit tests)

Package Concept – The rpact Manual



`help(package = "rpact")` : Inline help

Confirmatory Adaptive Clinical Trial Design and Analysis



Documentation for package 'rpact'

- [DESCRIPTION file.](#)
- [User guides, package vignettes and other documentation.](#)

Help Pages

rpact-package	rpact - Confirmatory Adaptive Clinical Trial Design and Analysis
getAccrualTime	Get Accrual Time
getAnalysisResults	Get Analysis Results
getAvailablePlotTypes	Get Available Plot Types
getClosedCombinationTestResults	Get Closed Combination Test Results
getClosedCombinationTestResultsEnrichment	Get Closed Combination Test Results
getClosedConditionalDunnettTestResults	Get Closed Conditional Dunnett Test Results
getConditionalPower	Get Conditional Power
getConditionalRejectionProbabilities	Get Conditional Rejection Probabilities
getData	Get Simulation Data
getDataset	Get Dataset
getDesignCharacteristics	Get Design Characteristics
getDesignConditionalDunnett	Get Design Conditional Dunnett Test
getDesignFisher	Get Design Fisher
getDesignGroupSequential	Get Design Group Sequential
getDesignInverseNormal	Get Design Inverse Normal
getDesignSet	Get Design Set
getEventProbabilities	Get Event Probabilities
getFinalConfidenceInterval	Get Final Confidence Interval
getFinalPValue	Get Final P Value

Package Concept – Most parameters have a default value

Example: `getDesignInverseNormal()` produces the output:

Design parameters and output of inverse normal combination test design:

User defined parameters: not available

Derived from user defined parameters: not available

Default parameters:

```
Type of design           : OF
Maximum number of stages : 3
Stages                   : 1, 2, 3
Information rates        : 0.333, 0.667, 1.000
Significance level       : 0.0250
Type II error rate       : 0.2
Two-sided power          : FALSE
Test                     : one-sided
Tolerance                : 1e-08
```

Output:

```
Cumulative alpha spending : 0.0002592, 0.0071601, 0.0250000
Critical values            : 3.471, 2.454, 2.004
Stage levels               : 0.0002592, 0.0070554, 0.0225331
```

Package Concept – Most parameters have a default value

Example: `getDesignInverseNormal(kMax = 2)` produces:

Design parameters and output of inverse normal combination test design:

User defined parameters:

```
Maximum number of stages : 2
Stages                    : 1, 2
```

Derived from user defined parameters:

```
Information rates        : 0.500, 1.000
```

Default parameters:

```
Type of design           : OF
Significance level       : 0.0250
Type II error rate      : 0.2
Two-sided power         : FALSE
Test                     : one-sided
Tolerance                : 1e-08
```

Output:

```
Cumulative alpha spending : 0.002583, 0.025000
Critical values           : 2.797, 1.977
Stage levels              : 0.002583, 0.023996
```

Sample Size and Power Calculation

Work-flow for sample size calculations in rpact

- 1 Define abstract group-sequential boundaries which are applicable to any type of endpoint (`getDesignGroupSequential()`).
- 2 Feed these boundaries into endpoint-specific sample size formulas (e.g., `getSampleSizeMeans()` , `getSampleSizeRates()` , `getSampleSizeSurvival()` , `getSimulationSurvival()`).

For trials without interim analyses, Step 1. can be omitted.

- 3 `getDesignInverseNormal()` yields the same results as `getDesignGroupSequential()` , it has an effect only for simulation and analysis.
- 4 `getDesignFisher()` provides no planning calculation, use the simulation tools instead.

Abstract group-sequential boundaries

- Function `getDesignGroupSequential()` derives group-sequential boundaries in the mathematically simplest case:
 - Single arm trial with independent $X_i \sim N(\mu, 1)$
 - Test $H_0 : \mu = 0$ against $H_1 : \mu = 1$
- Correlation structure between Z -statistics at interim and final analyses is identical for more complex situations (e.g., binary, continuous and survival endpoints).

Group-sequential boundaries and properties of the design apply to all endpoints!

Example: O'Brien-Fleming type α -spending

```
# Efficacy interim analyses at 30%, 60% and 100% information
design <- getDesignGroupSequential(
  sided = 2, alpha = 0.05, beta = 0.2,
  informationRates = c(0.3, 0.6, 1),
  typeOfDesign = "asOF")
```

- `informationRates` : information fractions at which interim and final analysis are conducted.
- Information fraction t_k at analysis k :
 - Binary and normal outcomes: $t_k = n_k / N_{max}$
 - Survival outcomes: $t_k = d_k / d_{max}$ where d is # events.
- `typeOfDesign = "asOF"` : O'Brien & Fleming type α -spending.

Supported efficacy boundaries

Argument `typeOfDesign` :

- Exact O'Brien & Fleming ("OF"), Pocock ("P"), Wang & Tsiatis ("WT"), Haybittle & Peto ("HP")
- Pampallona & Tsiatis ("PT") one-sided and two-sided designs
- O'Brien & Fleming and Pocock type α -spending ("asOF" and "asP")
- Kim & DeMets ("asKD") and Hwang, Shi & DeCani α -spending ("asHSD") and beta-spending ("bsKD" and "bsHSD"))
- User-defined α -spending ("asUser") and β -spending ("bsUser")
- No early efficacy stops ("noEarlyEfficacy")

Example: Futility boundaries

```
# Example: non-binding futility boundary at first interim in
# case estimated treatment effect is null or in "the wrong
# direction", no futility at second interim
design <- getDesignGroupSequential(
  sided = 1, alpha = 0.025, beta = 0.2,
  informationRates = c(0.3, 0.6, 1),
  typeOfDesign = "as0F",
  futilityBounds = c(0, -Inf),
  bindingFutility = FALSE)
```

- `futilityBounds`: Vector on z-value scale for interim analyses (excluding final analysis).
 - $z = 0$: Futility if “null effect or effect in wrong direction”
 - $z = -Inf$: No futility at this interim analysis
- `bindingFutility = FALSE` (default): no effect on efficacy boundaries.
- `futilityBounds` only supported for one-sided testing.

Output

```
print(design)
```

User defined parameters:

```
Type of design           : O'Brien & Fleming type alpha spending
Information rates        : 0.400, 0.800, 1.000
Futility bounds (non-binding) : 0, -Inf
```

Derived from user defined parameters:

```
Maximum number of stages : 3
```

Default parameters:

```
Stages                   : 1, 2, 3
Significance level       : 0.0250
Type II error rate       : 0.2000
Two-sided power          : FALSE
Binding futility         : FALSE
Test                     : one-sided
Tolerance                 : 1e-08
Type of beta spending    : none
```

Output:

```
Cumulative alpha spending : 0.0003942, 0.0122118, 0.0250000
Critical values            : 3.357, 2.255, 2.026
Stage levels (one-sided)  : 0.0003942, 0.0120779, 0.0213919
```

- **Critical values** : efficacy boundary values on z-value scale.
- **Stage levels** : local significance bounds.

Additional characteristics of the design

```
getDesignCharacteristics(design)
```

Group sequential design characteristics:

```
Number of subjects fixed      : 7.8
Shift                        : 8.1984
Inflation factor             : 1.0445
Informations                 : 3.279, 6.559, 8.198
Power                       : 0.06106, 0.61940, 0.80000
Rejection probabilities under H1 : 0.06106, 0.55835, 0.18060
Futility probabilities under H1  : 0.03508, 0
Ratio expected vs fixed sample size under H1 : 0.8676
Ratio expected vs fixed sample size under a value between H0 and H1 : 0.8927
Ratio expected vs fixed sample size under H0 : 0.7285
```

- **Number of subjects fixed** : for abstract design without interim analyses.
- **Shift** : Maximal sample size for abstract design with interim analyses.
- **Inflation factor** : Maximum sample size increase of sequential design relative to design without interim analyses.
- **Ratio expected vs fixed sample size** : Reduction in expected sample size of sequential relative to fixed design.

Stopping probabilities under H_0 and H_1

```
nMax <- getDesignCharacteristics(design)$shift
```

```
getPowerAndAverageSampleNumber(design,
  theta = 0, nMax = nMax)
```

```
getPowerAndAverageSampleNumber(design,
  theta = 1, nMax = nMax)
```

Output:

```
Average sample sizes (ASN) : 5.455
Power                       : 0.02344
Early stop                  : 0.5038
Early stop [1]             : 0.500043
Early stop [2]             : 0.003758
Early stop [3]             : NA
Overall reject              : 0.02344
Reject per stage [1]       : 4.273e-05
Reject per stage [2]       : 0.003758
Reject per stage [3]       : 0.01964
Overall futility           : 0.5000
Futility stop per stage [1] : 0.5000
Futility stop per stage [2] : 0.0000
```

Legend:

[k]: values at stage k

Output:

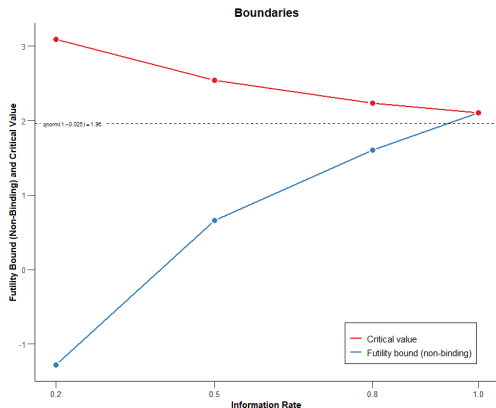
```
Average sample sizes (ASN) : 6.928
Power                       : 0.8000
Early stop                  : 0.3920
Early stop [1]             : 0.06572
Early stop [2]             : 0.32624
Early stop [3]             : NA
Overall reject              : 0.8000
Reject per stage [1]       : 0.009643
Reject per stage [2]       : 0.326241
Reject per stage [3]       : 0.464116
Overall futility           : 0.05607
Futility stop per stage [1] : 0.05607
Futility stop per stage [2] : 0.00000
```

Legend:

[k]: values at stage k

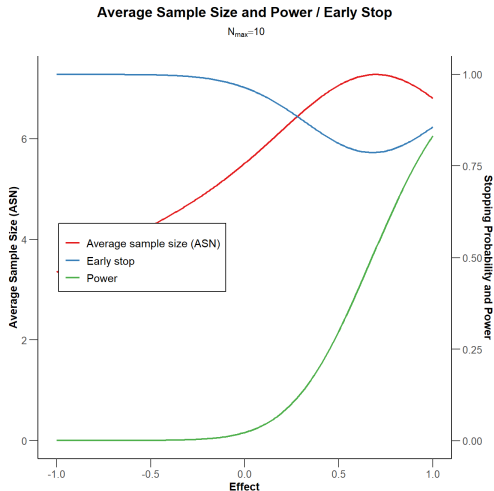
Example: Derivation of futility bounds

```
design <- getDesignInverseNormal(kMax = 4, alpha = 0.025,
  typeOfDesign = "asKD", gammaA = 2,
  informationRates = c(0.2, 0.5, 0.8, 1),
  typeBetaSpending = "bsOF",
  bindingFutility = FALSE)
plot(design, type = 1)
```



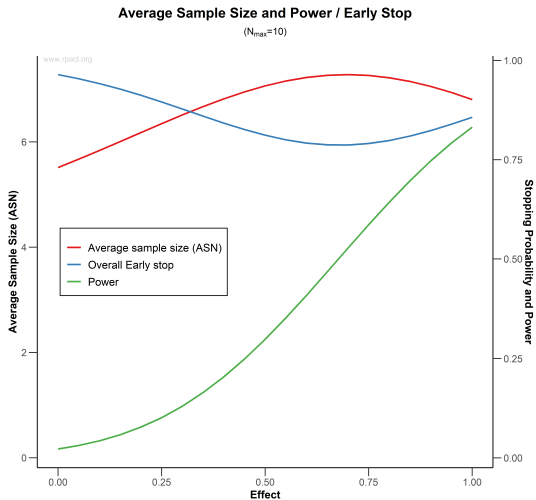
Example: Derivation of futility bounds

```
plot(design, type = 6, nMax = 10)
```



Example: Derivation of futility bounds

```
plot(design, type = 6, nMax = 10, theta = seq(0, 1, 0.05))
```



More on group-sequential boundaries

E.g., vignette “Defining group-sequential boundaries with `rpact`”, written by Marcel Wolbers.

Also contains information on:

- Extracting information from `rpact` objects
- β -spending functions for futility
- Plotting `rpact` objects

Sample Size Calculation for Continuous Endpoint

Exercise 2

Design without interim analyses

```
sampleSizeResult <- getSampleSizeMeans(  
  alternative = 10, stDev = 24, sided = 2,  
  alpha = 0.05, beta = 0.2,  
  allocationRatioPlanned = 2)
```

- `alternative` is the alternative hypothesis value. This can be a vector of assumed alternatives (default is `seq(0.2, 1, 0.2)`)
- `stDev` is the standard deviation (default is 1). If `meanRatio = TRUE` is specified, `stDev` defines the coefficient of variation `sigma/mu2`
- `allocationRatioPlanned` The planned allocation ratio for a two treatment groups design (default is 1);
e.g., `allocationRatioPlanned = 2 : 2(intervention) : 1(control)`
If `allocationRatioPlanned = 0` is entered, the optimal allocation ratio yielding the smallest overall sample size is determined

Design with interim analyses

```
# Design from above
design <- getDesignGroupSequential(
  sided = 1, alpha = 0.025, beta = 0.2,
  informationRates = c(0.3, 0.6, 1),
  typeOfDesign = "as0F",
  futilityBounds = c(0, -Inf),
  bindingFutility = FALSE)

# Sample size calculation
sampleSizeResult <- getSampleSizeMeans(
  design = design, alternative = 10, stDev = 24,
  allocationRatioPlanned = 2)
```

Design with interim analyses

```
summary(sampleSizeResult)
```

Sequential analysis with a maximum of 3 looks (group sequential design), overall significance level 2.5% (one-sided).

The sample size was calculated for a two-sample t-test, $H_0: \mu(1) - \mu(2) = 0$, $H_1: \text{effect} = 10$, standard deviation = 24, planned allocation ratio = 2, power 80%.

Stage	1	2	3
Information rate	30%	60%	100%
Efficacy boundary (z-value scale)	3.929	2.670	1.981
Futility boundary (z-value scale)	0	-Inf	
Overall power	0.0096	0.3359	0.8000
Expected number of subjects	181.3		
Number of subjects	66.0	132.1	220.1
Cumulative alpha spent	<0.0001	0.0038	0.0250
One-sided local significance level	<0.0001	0.0038	0.0238
Efficacy boundary (t)	26.286	12.016	6.836
Futility boundary (t)	0		
Overall exit probability (under H_0)	0.5000	0.0038	
Overall exit probability (under H_1)	0.0657	0.3262	
Exit probability for efficacy (under H_0)	<0.0001	0.0038	
Exit probability for efficacy (under H_1)	0.0096	0.3262	
Exit probability for futility (under H_0)	0.5000	0	
Exit probability for futility (under H_1)	0.0561	0	

Legend:

(t): treatment effect scale

Design with interim analyses

```
print(sampleSizeResult)
```

```

:
:
Number of subjects (1) [1]      : 44.0
Number of subjects (1) [2]      : 88.0
Number of subjects (1) [3]      : 146.7
Number of subjects (2) [1]      : 22.0
Number of subjects (2) [2]      : 44.0
Number of subjects (2) [3]      : 73.4
Expected number of subjects under H0      : 142.7
Expected number of subjects under H0/H1    : 181.8
Expected number of subjects under H1      : 181.3
Critical values (treatment effect scale) [1] : 26.286
Critical values (treatment effect scale) [2] : 12.016
Critical values (treatment effect scale) [3] : 6.836
Futility bounds (treatment effect scale) [1] : 0.000
Futility bounds (treatment effect scale) [2] : NA

```

Legend:

```
(i): values of treatment arm i
[k]: values at stage k
```

Critical values (treatment effect scale): Minimal detectable difference (MDD), i.e., smallest difference in observed means that would lead to a rejection at this stage (assuming observed standard deviation as specified.)

Sample Size Calculation for Binary Endpoint

Exercise 3

Planning of Survival Designs

Exercise 1 and bonus exercise 6

Simulation Functions

Simulation Functions

- Similar to power calculation, simulation tool available
- Fixed sample size or sample size recalculation can be assessed
- Very similar options as compared to power calculation functions for testing means, rates, and survival
- Survival simulation implemented in C++, so very fast
- Functions `getSimulationMeans()` ,
`getSimulationRates()` , and
`getSimulationSurvival()`

Example

Example

```
getSimulationMeans(plannedSubjects = 100)
```

User defined parameters:

```
Seed : -774025874
Planned cumulative subjects : 100
```

Default parameters:

```
Planned allocation ratio : 1
Maximum number of iterations : 1000
Standard deviation : 1
Alternatives : 0, 0.2, 0.4, 0.6, 0.8, 1
Treatment groups : 2
Direction upper : TRUE
Theta H0 : 0
Mean ratio : FALSE
Normal approximation : TRUE
```

Results:

```
Iterations : 1000, 1000, 1000, 1000, 1000, 1000
Overall reject : 0.0350, 0.1630, 0.5270, 0.8400, 0.9800, 0.9990
Reject per stage : 0.0350, 0.1630, 0.5270, 0.8400, 0.9800, 0.9990
Futility stop : 0, 0, 0, 0, 0, 0
Early stop : 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000
Expected number of subjects : 100.0, 100.0, 100.0, 100.0, 100.0, 100.0
Sample sizes : 100.0, 100.0, 100.0, 100.0, 100.0, 100.0
```

Example

```
getSimulationMeans(plannedSubjects = 100, showStatistics = TRUE)
```

Simulated data:

```
Number of subjects [1], alternative = 0      : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Number of subjects [1], alternative = 0.2    : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Number of subjects [1], alternative = 0.4    : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Number of subjects [1], alternative = 0.6    : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Number of subjects [1], alternative = 0.8    : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Number of subjects [1], alternative = 1      : median [range]: 100 [100 - 100]; mean +/-sd: 100 +/-0
Test statistic [1], alternative = 0          : median [range]: 0.081 [-3.236 - 3.414]; mean +/-sd: 0.078
Test statistic [1], alternative = 0.2        : median [range]: 0.944 [-2.727 - 4.012]; mean +/-sd: 0.96
Test statistic [1], alternative = 0.4        : median [range]: 2.033 [-1.377 - 5.147]; mean +/-sd: 2.018
Test statistic [1], alternative = 0.6        : median [range]: 3.026 [-0.2 - 6.95]; mean +/-sd: 3.029 +
Test statistic [1], alternative = 0.8        : median [range]: 3.966 [0.883 - 7.331]; mean +/-sd: 3.982
Test statistic [1], alternative = 1          : median [range]: 5.017 [1.676 - 8.095]; mean +/-sd: 4.991
```

Receive the data (i.e., test statistics etc., not raw data!) used for the simulation:

```
getData(getSimulationMeans(plannedSubjects = 100))
```

Example: Group Sequential Design

```
design <- getDesignGroupSequential()
getSimulationMeans(design, plannedSubjects = c(20, 40, 60))
```

Simulation of means (group sequential design):

```
:
:
```

Results:

```
Alternatives                : 0.0, 0.2, 0.4, 0.6, 0.8, 1.0
Iterations [1]              : 1000, 1000, 1000, 1000, 1000, 1000
Iterations [2]              : 1000, 996, 996, 986, 954, 903
Iterations [3]              : 994, 965, 881, 702, 466, 250
Overall reject               : 0.0240, 0.1110, 0.3450, 0.6540, 0.8670, 0.9670
Reject per stage [1]        : 0.0000, 0.0040, 0.0040, 0.0140, 0.0460, 0.0970
Reject per stage [2]        : 0.0060, 0.0310, 0.1150, 0.2840, 0.4880, 0.6530
Reject per stage [3]        : 0.0180, 0.0760, 0.2260, 0.3560, 0.3330, 0.2170
Futility stop per stage [1] : 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000
Futility stop per stage [2] : 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000
Futility stop                : 0, 0, 0, 0, 0
Early stop                   : 0.0060, 0.0350, 0.1190, 0.2980, 0.5340, 0.7500
Expected number of subjects  : 59.9, 59.2, 57.5, 53.8, 48.4, 43.1
Sample sizes [1]            : 20.0, 20.0, 20.0, 20.0, 20.0, 20.0
Sample sizes [2]            : 20.0, 20.0, 20.0, 20.0, 20.0, 20.0
Sample sizes [3]            : 20.0, 20.0, 20.0, 20.0, 20.0, 20.0
Conditional power (achieved) [1] : NA, NA, NA, NA, NA, NA
Conditional power (achieved) [2] : 0.0595, 0.1174, 0.2138, 0.3723, 0.5127, 0.6254
Conditional power (achieved) [3] : 0.0644, 0.1322, 0.2677, 0.4448, 0.5555, 0.6582
```

Simulation of Testing Means

```
getSimulationMeans(design, plannedSubjects, ...)
```

Returns the sample size for testing means in one and two samples.

- `design` The trial design.
- `groups` The number of treatment groups (1 or 2) (default is 2).
- `meanRatio` If `meanRatio = TRUE` is specified the sample size for one-sided testing of $H_0: \mu_1/\mu_2 = \theta_0$ is calculated (default is FALSE).
- `thetaH0` The null hypothesis value. For one-sided testing, a value $\neq 0$ (or a value $\neq 1$ for testing the mean ratio) can be specified (default is 0).
- `alternative` The alternative hypothesis value. This can be a vector of assumed alternatives (default is `seq(0.2, 1, 0.2)`).
- `stDev` The standard deviation (default is 1). If `meanRatio = TRUE` is specified, `stDev` defines the coefficient of variation σ/μ_2 .

Simulation of Testing Means

```
getSimulationMeans(design, plannedSubjects, ...)
```

- `plannedSubjects` `plannedSubjects` is a 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.
- `directionUpper` Specifies the direction of the alternative, only applicable for one-sided testing, default is `TRUE`.
- `allocationRatioPlanned` The planned allocation ratio for a two treatment groups design (default is 1).
- `maxNumberOfIterations` The number of simulation iterations.
- `seed` The seed to reproduce the simulation, default is a random seed.

Simulation of Testing Means

```
getSimulationMeans(design, plannedSubjects, ...)
```

- `conditionalPower` The conditional power under which the sample size recalculation is performed.
- `minNumberOfSubjectsPerStage` When performing a data driven sample size recalculation, the vector with length `kMax` `minNumberOfSubjectsPerStage` determines the minimum number of subjects per stage (i.e., not cumulated), the first element is not taken into account.
- `maxNumberOfSubjectsPerStage` Analogously
- `thetaH1` If specified, the value of the alternative under which the conditional power calculation is performed.
- `calcSubjectsFunction` Optionally, a function can be entered that defines the way of performing the sample size recalculation. By default, the sample size recalculation is performed with specified conditional power and `minNumberOfSubjectsPerStage` and `maxNumberOfSubjectsPerStage`.

Simulation of Testing Means

Example

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` .

```
designIN <- getDesignInverseNormal()  
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 = 1000)
```

Simulation of Testing Means

Example

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,
  thetaH1, stDevH1) {
  if (stage == 2) {
    stageSubjects <- 4 * (max(0,
      conditionalCriticalValue +
      qnorm(conditionalPower)))^2 /
      (max(1e-12, thetaH1 / stDevH1))^2
    stageSubjects <- min(max(
      minNumberOfSubjectsPerStage[stage], stageSubjects
    ), maxNumberOfSubjectsPerStage[stage])
  } else {
    stageSubjects <- sampleSizesPerStage[stage - 1]
  }
  return(stageSubjects)
}
```

Simulation of Testing Means

Example

```
getSimulationMeans(designIN, alternative = 2:4, stDev = 5,
  plannedSubjects = c(20, 40, 60),
  minNumberOfSubjectsPerStage = c(NA, 20, 20),
  maxNumberOfSubjectsPerStage = c(NA, 160, 160),
  conditionalPower = 0.8,
  calcSubjectsFunction = mySampleSizeCalculationFunction,
  maxNumberOfIterations = 1000)
```

- For testing rates, examples and sample size calculation formula can be found in `?getSimulationRates`
- Simulating rates: exercise 4
- Simulating survival: bonus exercise 7

Analysis with rpact

Current Methods

Analysing a Trial with Interim Stages

- Group sequential test
- Inverse normal combination test
- Fisher's combination test
- Repeated confidence intervals, p-Values
- Conditional power assessment
- Final analysis adjusted confidence intervals, p-Values
- Conditional Rejection Probability (Müller & Schäfer)
- All this for continuous, binary, and survival endpoint

Group Sequential Analysis

```
getAnalysisResults(design, dataInput, ...)
```

Given a design and a data set, 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 confidence intervals.)

The conditional power is calculated only if (at least) the sample size for the subsequent stage(s) is specified.

- `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`.

Group Sequential Analysis

dataInput

- An element of **DataSetMeans for one sample** is created by `getDataset(means = , stDevs = , sampleSizes =)` where `means`, `stDevs`, `sampleSizes` are vectors with stagewise means, standard deviations, and sample sizes of length given by the number of available stages.
- An element of **DataSetMeans for two samples** is created by `getDataset(means1 = , means2 = , stDevs1 = , stDevs2 = , sampleSizes1 = , sampleSizes2 =)` where `means1`, `means2`, `stDevs1`, `stDevs2`, `sampleSizes1`, `sampleSizes2` are vectors with stagewise means, standard deviations, and sample sizes 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(events = , sampleSizes =)` where `events`, `sampleSizes` are vectors with stagewise events and sample sizes of length given by the number of available stages.

Group Sequential Analysis

dataInput

- An element of `DataSetRates` for two samples is created by `getDataset(events1 =, events2 =, sampleSizes1 =, sampleSizes2 =)` where `events1`, `events2`, `sampleSizes1`, `sampleSizes2` are vectors with stagewise events and sample sizes 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 `allocationRatios` are the stagewise events, logrank statistics, and allocation ratios.

The data sets can also be created by importing raw data (e.g., from a SAS file), calculating estimated adjusted (marginal) means for a linear model (e.g., ANCOVA), and using the `emmeans` package to define the components in `getDataset()`.

Example

Specify design:

```
design <- getDesignInverseNormal(  
  kMax = 4, typeOfDesign = "WT", deltaWT = 0.45)
```

Data summary for binary data:

```
dataExample <- getDataset(  
  n1      = c( 8, 10,  9),  
  n2      = c(11, 13, 12),  
  events1 = c( 3,  4,  5),  
  events2 = c( 8, 10, 12))
```

Create results object:

```
results <- getAnalysisResults(design = design,  
  dataInput = dataExample, directionUpper = FALSE)
```

print(results)

```

Design parameters:
  Fixed weights                : 0.500, 0.500, 0.500, 0.500
  Critical values              : 2.456, 2.372, 2.325, 2.291
  Futility bounds (non-binding) : -Inf, -Inf, -Inf
  Cumulative alpha spending    : 0.007026, 0.013828, 0.019778, 0.025000
  Local one-sided significance levels : 0.007026, 0.008839, 0.010045, 0.010968
  Significance level           : 0.0250
  Test                         : one-sided
User defined parameters:
  Direction upper              : FALSE
Default parameters:
  Normal approximation         : TRUE
  Theta H0                    : 0
Stage results:
  Cumulative effect sizes      : -0.3523, -0.3611, -0.3889, NA
  Cumulative treatment rate    : 0.375, 0.389, 0.444, NA
  Cumulative control rate      : 0.727, 0.75, 0.833, NA
  Stage-wise test statistics    : -1.536, -1.799, -2.567, NA
  Stage-wise p-values          : 0.062328, 0.036037, 0.005133, NA
  Combination test statistics  : 1.536, 2.358, 3.407, NA
Analysis results:
  Actions                      : continue, continue, reject and stop, NA
  Conditional rejection probability : 0.07769, 0.30931, 0.90625, NA
  Conditional power            : NA, NA, NA, NA
  Repeated confidence intervals (lower) : -0.7386, -0.6456, -0.6185, NA
  Repeated confidence intervals (upper) : 0.197323, 0.002224, -0.140459, NA
  Repeated p-values           : 0.156147, 0.025923, 0.000906, NA
  Final stage                  : 3
  Final p-value                : NA, NA, 0.01387, NA
  Final CIs (lower)           : NA, NA, -0.5687, NA
  Final CIs (upper)           : NA, NA, -0.03726, NA
  Median unbiased estimate     : NA, NA, -0.3168, NA

```

summary(results)

Analysis results for a binary endpoint

Sequential analysis with 4 looks (inverse normal combination test design).
The results were calculated using a two-sample test for rates (one-sided),
normal approximation test.

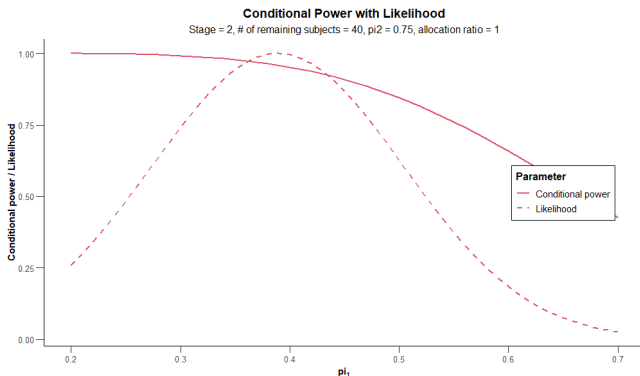
$H_0: \pi(1) - \pi(2) = 0$ against $H_1: \pi(1) - \pi(2) < 0$.

Stage	1	2	3	4
Fixed weight	0.5	0.5	0.5	0.5
Efficacy boundary (z-value scale)	2.456	2.372	2.325	2.291
Cumulative alpha spent	0.0070	0.0138	0.0198	0.0250
Stage level	0.0070	0.0088	0.0100	0.0110
Cumulative effect size	-0.352	-0.361	-0.389	
Cumulative treatment rate	0.375	0.389	0.444	
Cumulative control rate	0.727	0.750	0.833	
Stage-wise test statistic	-1.536	-1.799	-2.567	
Stage-wise p-value	0.0623	0.0360	0.0051	
Inverse normal combination	1.536	2.358	3.407	
Test action	continue	continue	reject and stop	
Conditional rejection probability	0.0777	0.3093	0.9062	
95% repeated confidence interval	[-0.739; 0.197]	[-0.646; 0.002]	[-0.618; -0.140]	
Repeated p-value	0.1561	0.0259	0.0009	
Final p-value			0.0139	
Final confidence interval			[-0.569; -0.037]	
Median unbiased estimate			-0.317	

Example

```
resultsStage2 <- getAnalysisResults(design, dataInput = dataExample,  
  stage = 2, pi1 = 0.45, pi2 = 0.75, nPlanned = c(20, 20),  
  directionUpper = FALSE)
```

```
plot(resultsStage2, piTreatmentRange = c(0.2, 0.7))
```



Example

```
plot(results, type = 2)
```

