Defining group-sequential boundaries with rpact

Marcel Wolbers, Gernot Wassmer, and Friedrich Pahlke

Last change: 20 Juni, 2019

In rpact, **sample size calculation for a group-sequential trial proceeds by following the same two steps regardless whether the endpoint is a continuous, binary, or a time-to-event endpoint**:

1. **Define the (abstract) group-sequential boundaries** of the design using the function `getDesignGroupSequential`.
2. **Calculate sample size for the endpoint of interest** by feeding the abstract boundaries from step 1. into specific functions for the endpoint of interest. This step uses functions such as `getSampleSizeMeans` (for continuous endpoints), `getSampleSizeRates` (for binary endpoints), and `getSampleSizeSurvival` (for survival endpoints).

The mathematical rationale for this two-step approach is that all group-sequential trials, regardless of the chosen endpoint type, rely on the fact that the Z-scores at different interim stages follow the same "canonical joint multivariate distribution" (at least asymptotically).

This R markdown file covers the more abstract first step and it gives example code for the the definition of the most commonly used group-sequential boundaries.

Step 2 is not covered in this document but it is covered in the separate endpoint-specific R markdown files for continuous, binary, and time to event endpoints. Of note, step 1. can be omitted for trials without interim analyses.

These examples are not intended to replace the official rpact documentation and help pages but rather to supplement them.

In general, rpact supports both one-sided and two-sided group-sequential designs. However, if futility boundaries are specified, only one-sided tests are permitted. **For simplicity, it is often preferred to use one-sided tests for group-sequential designs** (typically with $\alpha = 0.025$).

Load rpact package

```
# Load rpact
library(rpact)
packageVersion("rpact") # version should be version 2.0.1 or later
```

```
## [1] '2.0.1'
```

Designs with efficacy interim analyses

O'Brien&Fleming type α-spending

Example:

- Interim analyses at information fractions 33%, 67%, and 100% (`informationRates = c(0.33,0.67,1)`). [Note: For equally spaced interim analyses, one can also specify the maximum number of stages (`kMax`, including the final analysis) instead of the `informationRates`.]
- Lan&DeMets α-spending approximation to the O'Brien&Fleming boundary (`typeOfDesign="asOF"`)
- α-spending approaches allow for flexible timing of interim analyses and corresponding adjustment of boundaries.

```
design <- getDesignGroupSequential(sided = 1, alpha = 0.025,
                                   informationRates = c(0.33,0.67,1),
                                   typeOfDesign="asOF")
```

Standard O'Brien&Fleming boundary

The originally published O'Brien&Fleming boundary (`typeOfDesign="OF"`). Note that strict Type I error control is only guaranteed for standard boundaries without α-spending if the pre-defined interim schedule (i.e., the information fractions at which interim analyses are conducted) is exactly adhered to.

```
design <- getDesignGroupSequential(sided = 1, alpha = 0.025,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="OF")
```

Standard Pocock and Haybittle&Peto boundaries

Pocock (`typeOfDesign="P"` for standard boundary, `typeOfDesign="asP"` for α-spending version) or Haybittle&Peto (`typeOfDesign="HP"`) boundaries (reject at interim if Z-value exceeds 3).

```
design <- getDesignGroupSequential(sided = 1, alpha = 0.025,
    informationRates = c(0.33,0.67,1), typeOfDesign="P")
```

Other pre-defined boundaries

- Kim&DeMets α-spending (`typeOfDesign="asKD`) with parameter `gammaA` (power function: `gammaA=1` is linear spending, `gammaA=2` quadratic)
- Hwang, Shi & DeCani α-spending (`typeOfDesign="asHSD"`) with parameter `gammaA` (for details, see Wassmer&Brannath 2016, p. 76)
- Standard Wang&Tsiatis Delta classes (`typeOfDesign="WT"`) and (`typeOfDesign="WToptim"`)

```
# Quadratic Kim&DeMets alpha-spending
design <- getDesignGroupSequential(
    sided = 1, alpha = 0.025,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="asKD",gammaA=2)
```

User-defined α-spending functions

User-defined α-spending functions (`typeOfDesign="asUser"`) can be provided via argument `userAlphaSpending` which must conatin a numeric vector with elements $0 < \alpha_1 < \ldots < \alpha_{Kmax} = \alpha$ that define the values of the cumulative alpha-spending function at each interim analysis.

```
# Example: User-defined alpha-spending function which is very conservative at first
# interim (spend alpha=0.001), conservative at second (spend an additional
# alpha=0.01, i.e. total cumulative alpha spent is 0.011 up to second interim),
# and spends the remaining alpha at the final analysis (i.e. cumulative alpha=0.025)
design <- getDesignGroupSequential(sided = 1, alpha = 0.025,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="asUser",
    userAlphaSpending=c(0.001,0.01+0.001,0.025))
# $stageLevels below extract local significance levels across interim analyses.
# Note that the local significance level is exactly 0.001 at the first
# interim, but slightly >0.01 at the second interim because the design
# exploits correlations between interim analyses.
design$stageLevels
```

```
## [1] 0.00100000 0.01052883 0.02004781
```

Designs with efficacy and futility interim analyses

Futility boundary at interims manually defined on Z-scale

- Argument `futilityBounds` contains a vector of futility bounds (on the Z-value scale) for each interim (but not the final analysis).
- A futility bounds of $Z = 0$ corresponds to an estimated treatment effect of zero or "null", i.e., in this case futility stopping is recommended if the treatment effect estimate at the interim analysis is zero or "goes in the wrong direction". Futility bounds of $Z = -\infty$ (which are numerically equivalent to $Z = -6$) correspond to no futility stopping at an interim.
- Due to the design of rpact, it is not possible to directly define futility boundaries on the treatment effect scale. If this is desired, one would need to manually convert the treatment effect scale to the Z-scale or, alternatively, experiment by varying the boundaries on the Z-scale until this implies the targeted critical values on the treatment effect scale. (Critical values on treatment effect scales are routinely provided by sample size functions for different endpoint types such as `getSampleSizeMeans` (for continuous endpoints), `getSampleSizeRates` (for binary endpoints), and `getSampleSizeSurvival` (for survival endpoints). Please see the R markdown files for these endpoint types for further details.)
- By default, all futility boundaries are non-binding (`bindingFutility=FALSE`). Binding futility boundaries (`bindingFutility=TRUE`) are not recommended.

```
# Example: non-binding futility boundary at each interim in case
# estimated treatment effect is null or goes in "the wrong direction"
design <- getDesignGroupSequential(sided = 1, alpha = 0.025,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="asOF",
    futilityBounds=c(0,0),
    bindingFutility = FALSE)
```

Formal β-spending

Formal β-spending functions is defined in the same way as for α-spending functions, e.g., Pocock type β-spending can be specified as `typeBetaSpending="bsP"` and `beta` needs to be specified, the default is `beta = 0.20`. Note that in the case, due to compuational constraints, the futility bounds are always non-binding (`bindingFutility = FALSE` produces an error message).

```
# Example: non-binding futility boundary at each interim in case
# estimated treatment effect is null or goes in "the wrong direction"
design <- getDesignGroupSequential(sided = 1, alpha = 0.025, beta = 0.1,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="asOF",
    typeBetaSpending="bsP")
```

Designs with futility interim analyses only

Such designs can be implemented by using a user-defined α-spending function which spends all of the type I error at the final analysis. Note that such designs do not allow stopping for efficacy regardless how persuasive the effect is.

```
# Example: non-binding futility boundary using an O'Brien & Fleming type
# beta spending function. No early stopping for efficacy (i.e. all alpha
# is spent at the final analysis).
design <- getDesignGroupSequential(sided = 1, alpha = 0.025, beta = 0.2,
```

```
    informationRates = c(0.33, 0.67, 1),
    typeOfDesign = "asUser",
    userAlphaSpending = c(0, 0, 0.025),
    typeBetaSpending = "bsOF",
    bindingFutility = FALSE)
```

Interpreting, accessing and printing rpact boundary objects

Information contained in the `design` object

```
design <- getDesignGroupSequential(sided = 1, alpha = 0.025, beta = 0.2,
    informationRates = c(0.33,0.67,1),
    typeOfDesign="asOF",
    futilityBounds=c(0,0),
    bindingFutility = FALSE)
design
```

```
## Design parameters and output of group sequential design:
##
## User defined parameters:
##   Type of design                          : asOF
##   Information rates                       : 0.330, 0.670, 1.000
##   Futility bounds (non-binding)           : 0.000, 0.000
##
## Derived from user defined parameters:
##   Maximum number of stages                : 3
##
## Default parameters:
##   Stages                                  : 1, 2, 3
##   Significance level                      : 0.0250
##   Type II error rate                      : 0.2
##   Two-sided power                         : FALSE
##   Binding futility                        : FALSE
##   Test                                    : one-sided
##   Tolerance                               : 1e-08
##   Type of beta spending                   : none
##
## Output:
##   Cumulative alpha spending               : 9.549e-05, 6.176e-03, 2.500e-02
##   Critical values                         : 3.731, 2.504, 1.994
##   Stage levels                            : 9.549e-05, 6.142e-03, 2.309e-02
```

Key information is contained in the object including **critical values on the Z-scale** ("critical values" in rpact output, `design$criticalValues`) and **one-sided local significance levels** (**"stage levels"** in rpact output,`design$stageLevels`). Note that the local significance levels are always given as one-sided levels in rpact even if a two-sided design is specified.

`names(design)` provides names of all objects included in the `design` object and `as.data.frame(design)` collects all design information into one data frame. `summary(design)` gives a slightly more detailed output. For more details about applying R generics to rpact objects, please refer to the separte R markdown file dedicated to this topic.

```
names(design)
```

```
##  [1] "kMax"                  "alpha"
##  [3] "stages"                "informationRates"
##  [5] "userAlphaSpending"     "criticalValues"
##  [7] "stageLevels"           "alphaSpent"
##  [9] "bindingFutility"       "tolerance"
## [11] "typeOfDesign"          "beta"
## [13] "deltaWT"               "futilityBounds"
## [15] "gammaA"                "gammaB"
## [17] "optimizationCriterion" "sided"
## [19] "betaSpent"             "typeBetaSpending"
## [21] "userBetaSpending"      "power"
## [23] "twoSidedPower"         "constantBoundsHP"
```

Stopping probabilities and expected sample size reduction

`getDesignCharacteristics(design)` provides more detailed information about the design:

```
designChar <- getDesignCharacteristics(design)
designChar
```

```
## Group sequential design characteristics:
##   Number of subjects fixed                      : 7.8
##   Shift                                         : 8.3241
##   Inflation factor                              : 1.0605
##   Informations                                  : 2.7469, 5.5771, 8.3241
##   Power                                         : 0.01907, 0.44296, 0.80000
##   Rejection probabilities                       : 0.01907, 0.42389, 0.35704
##   Futility probabilities                        : 0.048720, 0.003437
##   Ratio expected vs fixed sample size under H1 : 0.8628
##   Ratio expected vs fixed sample size under a value between H0 and H1 : 0.8689
##   Ratio expected vs fixed sample size under H0 : 0.6589
```

```
names(designChar)
```

```
##  [1] "nFixed"                 "shift"
##  [3] "inflationFactor"        "stages"
##  [5] "information"            "power"
##  [7] "rejectionProbabilities" "futilityProbabilities"
##  [9] "averageSampleNumber1"   "averageSampleNumber01"
## [11] "averageSampleNumber0"
```

Note that the design characteristics despend on beta that needs to be specified in getDesignGroupSequential. By default, beta = 0.20.

Explanations regarding the output:

- **Maximum sample size inflation factor** (`$inflationFactor`): This is the maximal sample size a group-sequential trial requires relative to the sample size of a fixed design without interim analyses.
- Probabilities of stopping due to a significant result at each interim or the final analysis (`$rejectionProbabilities`), cumulative power (`$power`), and probability of stopping for futility at each interim (`$futilityProbabilities`). All of these are calculated under the alternative H1.
- **Expected sample size** of group-sequential design (relative to fixed design) under the alternative hypothesis H1 (`$averageSampleNumber1`), under the null hypothesis H0 (`$averageSampleNumber0`), and under the parameter in the middle between H0 and H1.
- In addition, `getDesignCharacteristics(design)` provides the required sample size for an abstract group-sequential single arm trial with a normal outcome, effect size 1, and standard deviation 1 (i.e. the

simplest group-sequential setting from a mathematical point of view). The sample size for such a trial without interim analyses is given as `$nFixed` and the maximum sample size of the corresponding group-sequential design as `$shift`.

The practical relevance of this abstract design is that the **properties of the design** (critical values, sample size inflation factor, rejection probabilies, etc) **carry over to group-sequential designs regardless of the endpoint (e.g. continuous, binary, or survial)** as they all share the same underlying canonical multivariate normal distribution of the Z-scores.

Overall stopping probabilities, rejection probabilities, and futility probabilities under the null (H0) and the alternative (H1) (overall and at each stage) can be calculated using the function `getPowerAndAverageSampleNumber`. To get these numbers, one needs to provide the maximum sample size and the effect size (0 under H0, 1 under H1) of the corresponding type of design.

```
# theta=0 for calculations under H0
getPowerAndAverageSampleNumber(design,theta=c(0),nMax=designChar$shift)
```

```
## Power and average sample size (ASN):
##
## User defined parameters:
##   N_max                                  : 8.32406460744414
##   Effect                                 : 0
##
## Default parameters: not available
##
## Output:
##   Average sample sizes (ASN)             : 5.17
##   Power                                  : 0.02377
##   Overall Early stop                     : 0.632
##   Early stop [1]                         : 0.500
##   Early stop [2]                         : 0.132
##   Early stop [3]                         : NA
##   Overall reject                         : 0.0238
##   Reject per stage [1]                   : 9.55e-05
##   Reject per stage [2]                   : 6.06e-03
##   Reject per stage [3]                   : 1.76e-02
##   Overall futility                       : 0.626
##   Futility stop per stage [1]            : 0.500
##   Futility stop per stage [2]            : 0.126
##
## Legend:
##   [k]: values at stage k
```

```
# theta=1 for calculations under alternative H1
getPowerAndAverageSampleNumber(design,theta=1,nMax=designChar$shift)
```

```
## Power and average sample size (ASN):
##
## User defined parameters:
##   N_max                                  : 8.32406460744414
##   Effect                                 : 1
##
## Default parameters: not available
##
## Output:
##   Average sample sizes (ASN)             : 6.77
##   Power                                  : 0.8000
```

```
##   Overall Early stop                          : 0.495
##   Early stop [1]                              : 0.0678
##   Early stop [2]                              : 0.4273
##   Early stop [3]                              : NA
##   Overall reject                              : 0.8
##   Reject per stage [1]                        : 0.0191
##   Reject per stage [2]                        : 0.4239
##   Reject per stage [3]                        : 0.3570
##   Overall futility                            : 0.0522
##   Futility stop per stage [1]                 : 0.04872
##   Futility stop per stage [2]                 : 0.00344
##
## Legend:
##   [k]: values at stage k
```

Note that the power under H0, i.e. the significance level, is slightly below 0.025 in this example as it is calculated under the assumption that the non-binding futility boundaries are adhered to.

Both (and even more) values can be obtained with one command `getPowerAndAverageSampleNumber(design,theta=c(0,1),r`

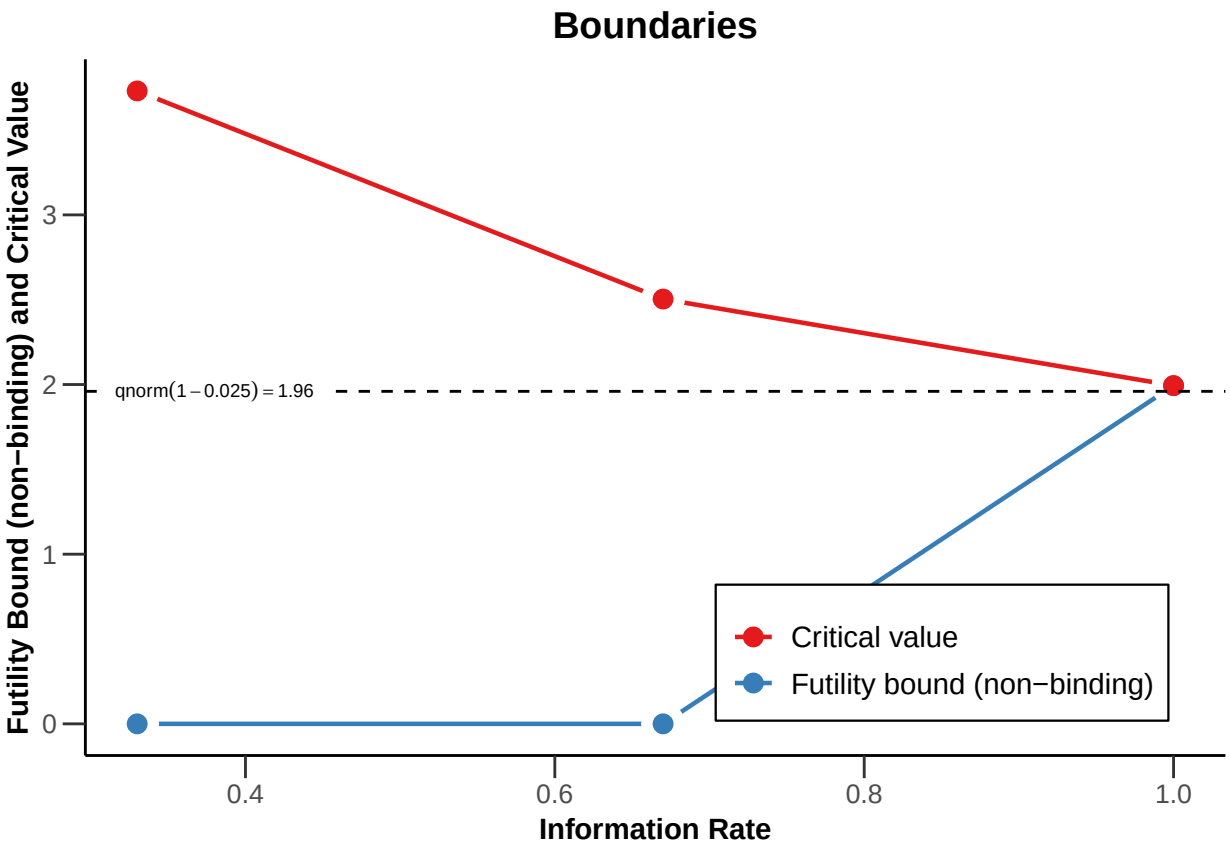
Plotting rpact boundary objects

Boundaries can be plotted using the `plot` (or `plot.TrialDesign`) function which produces a ggplot2 object.

The most relevant plots for (abstract) boundaries without easily interpretable treatment effect are boundary plots on Z-scale (`type=1`) or p-value-scale (`type=3`) as well as plots of the α-spending function (`type=4`). Conveniently, argument `showSource=TRUE`also provides the source data for the plot. For examples of all available plots, see the R markdown file How to create admirable plots with rpact.

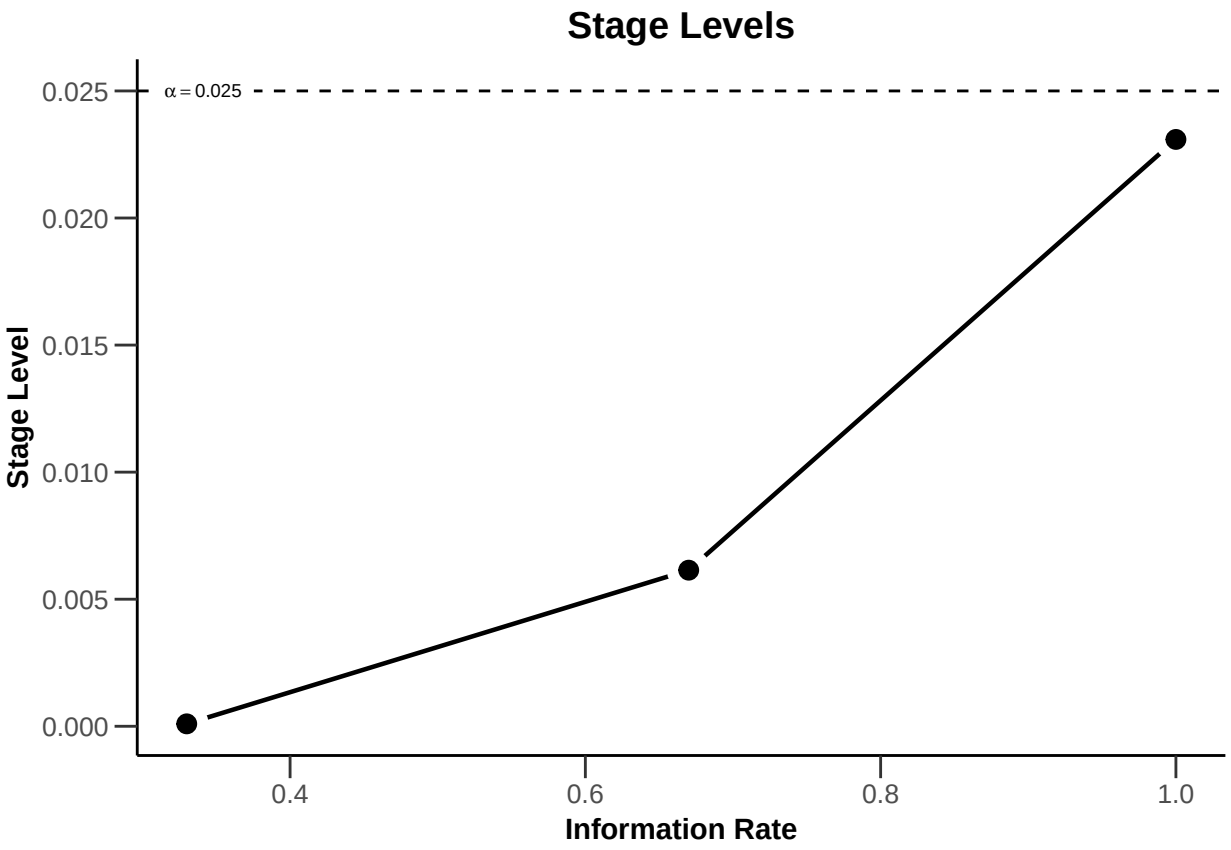
```
plot(design,type=1,showSource=TRUE)
```

```
## Source data of the plot:
##   x-axis: design$informationRates
##   y-axes:
##     y1: design$futilityBounds
##     y2: design$criticalValues
```



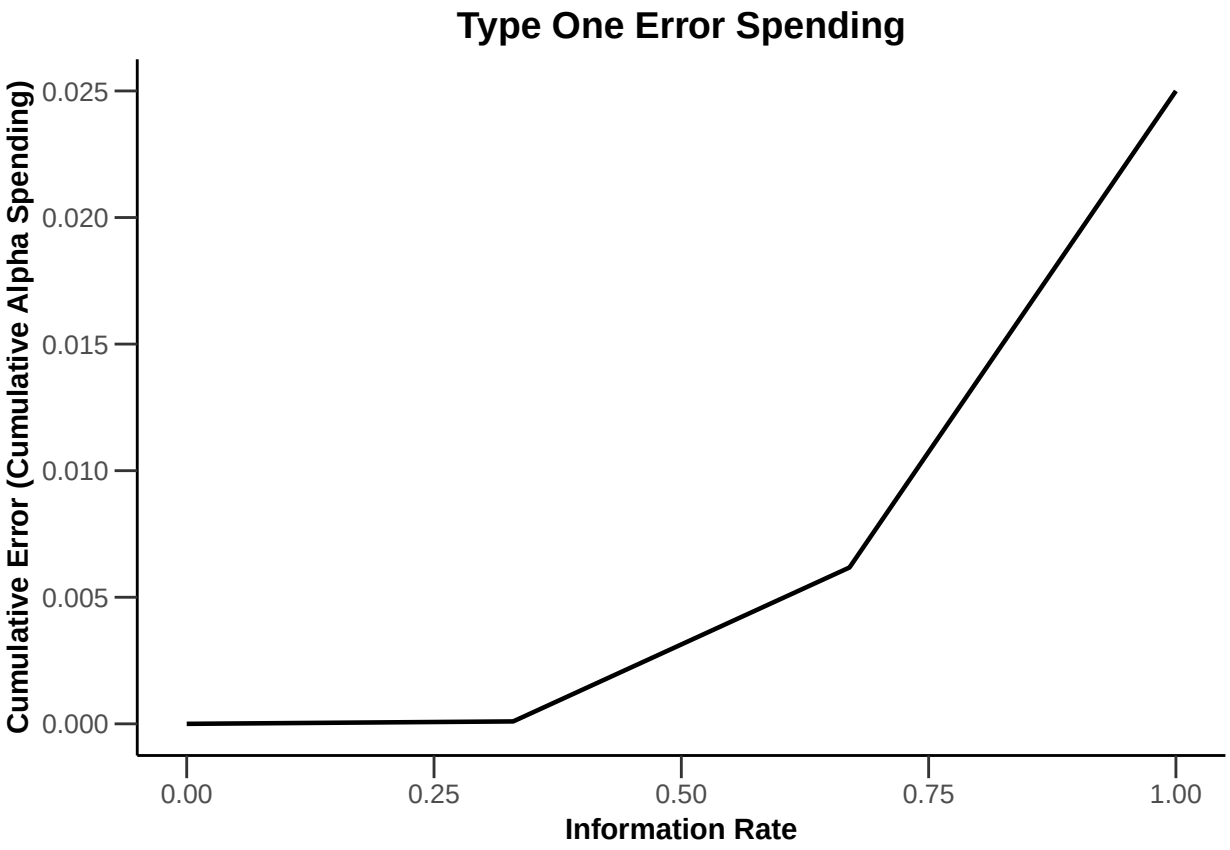

```
plot(design,type=3,showSource=TRUE)
```

```
## Source data of the plot:
##   x-axis: design$informationRates
##   y-axis: design$stageLevels
```




```
plot(design,type=4,showSource=TRUE)
```

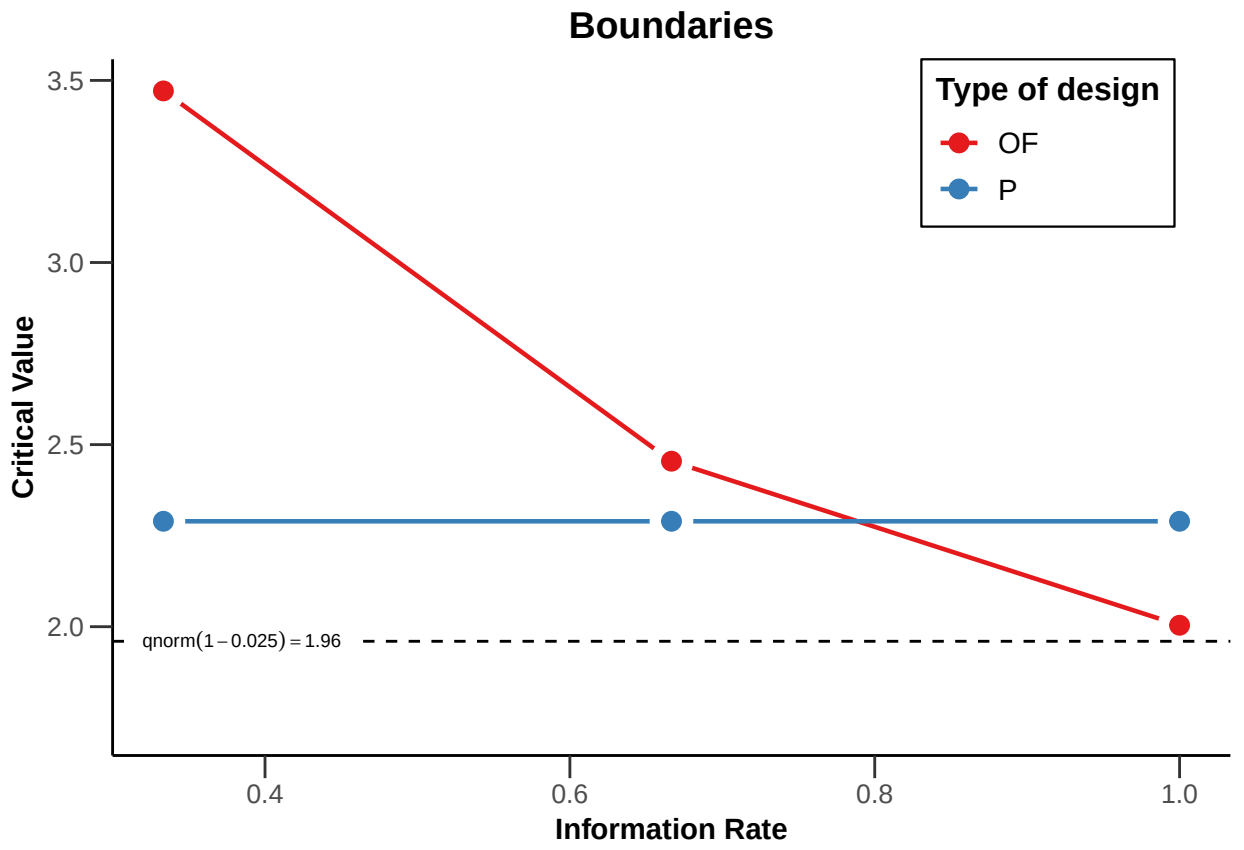
```
## Source data of the plot:
##   x-axis: design$informationRates
##   y-axis: design$alphaSpent
```



Comparison of multiple designs

Multiple designs can be combined into a design set (`getDesignSet`) and their properties plotted jointly:

```
#O'Brien&Fleming, 3 equally spaced stages
d1 <- getDesignGroupSequential(typeOfDesign = "OF",kMax=3)
# Pocock
d2 <- getDesignGroupSequential(typeOfDesign = "P",kMax=3)
designSet <-getDesignSet(designs = c(d1, d2), variedParameters = "typeOfDesign")
plot(designSet,type=1)
```



System: rpact 2.0.1, R version 3.5.2 (2018-12-20), platform: x86_64-w64-mingw32

To cite package 'rpact' in publications use:

Gernot Wassmer and Friedrich Pahlke (2019). rpact: Confirmatory Adaptive Clinical Trial Design and Analysis. R package version 2.0.1. https://CRAN.R-project.org/package=rpact

License