An Example to Illustrate Boundary Re-Calculations during the Trial with rpact

Marcel Wolbers, Gernot Wassmer, and Friedrich Pahlke

Last change: 15 November, 2022

Contents

Summary

This R Markdown document provides an example for updating the group sequential boundaries when using an α-spending function approach based on observed information rates in rpact. Since version 3.1 of rpact, an additional option in the `getAnalysisResults()` function provides an easy way to perform an analysis with critical values that are calculated subsequently during the stages of the trial.

1 Introduction

Group-sequential designs based on α-spending functions protect the Type I error exactly even if the pre-planned interim schedule is not exactly adhered to. However, this requires re-calculation of the group sequential boundaries at each interim analysis based on actually observed information fractions. Unless deviations from the planned information fractions are substantial, the re-calculated boundaries are quite similar to the pre-planned boundaries and the re-calculation will affect the actual test decision only on rare occasions.

Importantly, it is not allowed that the timing of future interim analyses is "motivated" by results from earlier interim analyses as this could inflate the Type I error rate. Deviations from the planned information fractions should thus only occur due to operational reasons (as it is difficult to hit the exact number of events exactly in a real trial) or due to external evidence.

The general principles for these boundary re-calculation are as follows (see also, Wassmer & Brannath, 2016, p78f):

- Updates at interim analyses prior to the final analysis:
  - Information fractions are updated according to the actually observed information fraction at the interim analysis relative to the **planned** maximum information.
  - The planned α-spending function is then applied to these updated information fractions.
- Updates at the final analysis in case the observed information at the final analysis is larger ("over-running") or smaller ("under-running") than the planned maximum information:
  - Information fractions are updated according to the actually observed information fraction at all interim analyses relative to the **observed** maximum information. $\Rightarrow$ Information fraction at final analysis is re-set to 1 but information fractions for earlier interim analyses are also changed.
  - The originally planned α-spending function cannot be applied to these updated information fractions because this would modify the critical boundaries of earlier interim analyses which is clearly not allowed. Instead, one uses the α that has actually been spent at earlier interim analyses and spends all remaining α at the final analysis.

This general principle be implemented via a user-defined α-spending function and is illustrated for an example trial with a survival endpoint below. We provide two solutions to the problem: the first is a way how existing tools in rpact can directly be used to solve the problem, the second is an automatic recalculation of the boundaries using a new parameter set (`maxInformation` and `informationEpsilon`) which is available in the `getAnalysisResults()` function since rpact version 3.1. This solution is described in Section 7 at the end of this document.

First, load the rpact package

```
library(rpact)
packageVersion("rpact") # version should be version 3.1 or later
```

```
## [1] '3.3.2'
```

2 Original trial design

The original trial design for this example is based on a standard O'Brien & Fleming type α-spending function with planned efficacy interim analyses after 50% and 75% of information as specified below.

```
# Initial design
design <- getDesignGroupSequential(
    sided = 1, alpha = 0.025, beta = 0.2,
    informationRates = c(0.5, 0.75, 1), typeOfDesign = "asOF"
)

# Initial sample size calculation
sampleSizeResult <- getSampleSizeSurvival(
    design = design,
    lambda2 = log(2) / 60, hazardRatio = 0.75,
    dropoutRate1 = 0.025, dropoutRate2 = 0.025, dropoutTime = 12,
    accrualTime = 0, accrualIntensity = 30,
    maxNumberOfSubjects = 1000
)

# Summarize design
kable(summary(sampleSizeResult))
```

Sample size calculation for a survival endpoint

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 logrank test, H0: hazard ratio = 1, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12, power 80%.

Stage	1	2	3
Information rate	50%	75%	100%
Efficacy boundary (z-value scale)	2.963	2.359	2.014
Overall power	0.1680	0.5400	0.8000
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0
Cumulative number of events	193.4	290.1	386.8
Analysis time	39.1	52.7	69.1
Expected study duration	58.0		
Cumulative alpha spent	0.0015	0.0096	0.0250
One-sided local significance level	0.0015	0.0092	0.0220
Efficacy boundary (t)	0.653	0.758	0.815
Exit probability for efficacy (under H0)	0.0015	0.0081	
Exit probability for efficacy (under H1)	0.1680	0.3720	

Legend:

- *(t)*: treatment effect scale

3 Boundary and power update at the first interim analysis

Assume that the first interim was conducted after 205 rather than the planned 194 events.

The updated design is calculated as per the code below. Note that for the calculation of boundary values on the treatment effect scale, we use the function `getPowerSurvival()` with the updated design rather than the function `getSampleSizeSurvival()` as we are only updating the boundary, not the sample size or the maximum number of events.

```
# Update design using observed information fraction at first interim.
# Information fraction of later interim analyses is not changed.
designUpdate1 <- getDesignGroupSequential(
    sided = 1, alpha = 0.025, beta = 0.2,
    informationRates = c(205 / 387, 0.75, 1), typeOfDesign = "asOF"
)

# Recalculate the power to get boundary values on the effect scale
# (Use original maxNumberOfEvents and sample size)
powerUpdate1 <- getPowerSurvival(
    design = designUpdate1,
    lambda2 = log(2) / 60, hazardRatio = 0.75,
    dropoutRate1 = 0.025, dropoutRate2 = 0.025, dropoutTime = 12,
    accrualTime = 0, accrualIntensity = 30,
    maxNumberOfSubjects = 1000, maxNumberOfEvents = 387, directionUpper = FALSE
)
```

The updated information rates and corresponding boundaries as per the output above are summarized as follows:

Power calculation for a survival endpoint

Sequential analysis with a maximum of 3 looks (group sequential design), overall significance level 2.5% (one-sided). The results were calculated for a two-sample logrank test, H0: hazard ratio = 1, power directed towards smaller values, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, maximum number of events = 387, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12.

Stage	1	2	3
Information rate	53%	75%	100%
Efficacy boundary (z-value scale)	2.867	2.366	2.015
Overall power	0.2097	0.5391	0.8001
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0
Expected number of events	317.0		
Cumulative number of events	205.0	290.2	387.0
Analysis time	40.6	52.7	69.1
Expected study duration	57.8		
Cumulative alpha spent	0.0021	0.0096	0.0250
One-sided local significance level	0.0021	0.0090	0.0220
Efficacy boundary (t)	0.670	0.758	0.815
Exit probability for efficacy (under H0)	0.0021	0.0076	
Exit probability for efficacy (under H1)	0.2097	0.3294	

Legend:

- *(t)*: treatment effect scale

4 Boundary and power update at the second interim analysis

Assume that the efficacy boundary was not crossed at the first interim analysis and the trial continued to the second interim analysis which was conducted after 285 rather than the planned 291 events. The updated design is calculated in the same way as for the first interim analysis as per the code below. The idea is to use the cumulative α spent from the first stage and an updated cumulative α that is spent for the second stage. For the second stage, this can be obtained with the original O'Brien & Fleming α-spending function:

```
# Update design using observed information fraction at first and second interim.
designUpdate2 <- getDesignGroupSequential(
    sided = 1, alpha = 0.025, beta = 0.2,
    informationRates = c(205 / 387, 285 / 387, 1), typeOfDesign = "asOF"
)

# Recalculate power to get boundary values on effect scale
# (Use original maxNumberOfEvents and sample size)
powerUpdate2 <- getPowerSurvival(
    design = designUpdate2,
    lambda2 = log(2) / 60, hazardRatio = 0.75,
    dropoutRate1 = 0.025, dropoutRate2 = 0.025, dropoutTime = 12,
    accrualTime = 0, accrualIntensity = 30,
    maxNumberOfSubjects = 1000, maxNumberOfEvents = 387, directionUpper = FALSE
)
kable(summary(powerUpdate2))
```

Power calculation for a survival endpoint

Sequential analysis with a maximum of 3 looks (group sequential design), overall significance level 2.5% (one-sided). The results were calculated for a two-sample logrank test, H0: hazard ratio = 1, power directed towards smaller values, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, maximum number of events = 387, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12.

Stage	1	2	3
Information rate	53%	73.6%	100%
Efficacy boundary (z-value scale)	2.867	2.393	2.011
Overall power	0.2097	0.5198	0.8004
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0
Expected number of events	317.2		
Cumulative number of events	205.0	285.0	387.0
Analysis time	40.6	51.9	69.1
Expected study duration	57.8		
Cumulative alpha spent	0.0021	0.0090	0.0250
One-sided local significance level	0.0021	0.0084	0.0222
Efficacy boundary (t)	0.670	0.753	0.815
Exit probability for efficacy (under H0)	0.0021	0.0069	
Exit probability for efficacy (under H1)	0.2097	0.3101	

Legend:

- *(t)*: treatment effect scale

5 Boundary and power update at the final analysis

Assume that the efficacy boundary was also not crossed at the second interim analysis and the trial continued to the final analysis which was conducted after 393 rather than the planned 387 events. The updated design is calculated as per the code below. The idea here to use the cumulative α spent from the first *and* the second stage stage and the final α that is spent for the last stage. An updated correlation has to be used and the original O'Brien & Fleming α-spending function cannot be used anymore. Instead, the α-spending function needs to be user defined as follows:

```
# Update boundary with information fractions as per actually observed event numbers
# !! use user-defined alpha-spending and spend alpha according to actual alpha spent
#    according to the second interim analysis
designUpdate3 <- getDesignGroupSequential(
    sided = 1, alpha = 0.025, beta = 0.2,
    informationRates = c(205, 285, 393) / 393,
    typeOfDesign = "asUser",
    userAlphaSpending = designUpdate2$alphaSpent
)

# Recalculate power to get boundary values on effect scale
# (Use planned sample size and **observed** maxNumberOfEvents)
powerUpdate3 <- getPowerSurvival(
    design = designUpdate3,
    lambda2 = log(2) / 60, hazardRatio = 0.75,
    dropoutRate1 = 0.025, dropoutRate2 = 0.025, dropoutTime = 12,
    accrualTime = 0, accrualIntensity = 30,
    maxNumberOfSubjects = 1000, maxNumberOfEvents = 393, directionUpper = FALSE
```

```
)
kable(summary(powerUpdate3))
```

Power calculation for a survival endpoint

Sequential analysis with a maximum of 3 looks (group sequential design), overall significance level 2.5% (one-sided). The results were calculated for a two-sample logrank test, H0: hazard ratio = 1, power directed towards smaller values, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, maximum number of events = 393, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12.

Stage	1	2	3
Information rate	52.2%	72.5%	100%
Efficacy boundary (z-value scale)	2.867	2.393	2.014
Overall power	0.2097	0.5198	0.8060
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0
Expected number of events	320.1		
Cumulative number of events	205.0	285.0	393.0
Analysis time	40.6	51.9	70.3
Expected study duration	58.4		
Cumulative alpha spent	0.0021	0.0090	0.0250
One-sided local significance level	0.0021	0.0084	0.0220
Efficacy boundary (t)	0.670	0.753	0.816
Exit probability for efficacy (under H0)	0.0021	0.0069	
Exit probability for efficacy (under H1)	0.2097	0.3101	

Legend:

- *(t)*: treatment effect scale

6 Overview of all updates

For easier comparison, all discussed boundary updates and power calculations are summarized more conveniently below. Note that each update only affects boundaries for the current or later analyses, i.e., earlier boundaries are never retrospectively modified.

6.1 Original design

Sample size calculation for a survival endpoint

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 logrank test, H0: hazard ratio = 1, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12, power 80%.

Stage	1	2	3
Information rate	50%	75%	100%
Efficacy boundary (z-value scale)	2.963	2.359	2.014
Overall power	0.1680	0.5400	0.8000
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0

Stage	1	2	3
Cumulative number of events	193.4	290.1	386.8
Analysis time	39.1	52.7	69.1
Expected study duration	58.0		
Cumulative alpha spent	0.0015	0.0096	0.0250
One-sided local significance level	0.0015	0.0092	0.0220
Efficacy boundary (t)	0.653	0.758	0.815
Exit probability for efficacy (under H0)	0.0015	0.0081	
Exit probability for efficacy (under H1)	0.1680	0.3720	

Legend:

- *(t)*: treatment effect scale

6.2 Updated boundaries and power at the first, second, and final analysis

Power calculation for a survival endpoint

Sequential analysis with a maximum of 3 looks (group sequential design), overall significance level 2.5% (one-sided). The results were calculated for a two-sample logrank test, H0: hazard ratio = 1, power directed towards smaller values, H1: hazard ratio = 0.75, control lambda(2) = 0.012, maximum number of subjects = 1000, maximum number of events = 393, accrual time = 33.333, accrual intensity = 30, dropout rate(1) = 0.025, dropout rate(2) = 0.025, dropout time = 12.

Stage	1	2	3
Information rate	52.2%	72.5%	100%
Efficacy boundary (z-value scale)	2.867	2.393	2.014
Overall power	0.2097	0.5198	0.8060
Expected number of subjects	1000.0		
Number of subjects	1000.0	1000.0	1000.0
Expected number of events	320.1		
Cumulative number of events	205.0	285.0	393.0
Analysis time	40.6	51.9	70.3
Expected study duration	58.4		
Cumulative alpha spent	0.0021	0.0090	0.0250
One-sided local significance level	0.0021	0.0084	0.0220
Efficacy boundary (t)	0.670	0.753	0.816
Exit probability for efficacy (under H0)	0.0021	0.0069	
Exit probability for efficacy (under H1)	0.2097	0.3101	

Legend:

- *(t)*: treatment effect scale

7 Automatic boundary recalculation at analysis stage

We now show how a concrete data analysis with an α-spending function design can be performed by specifying the parameter `maxInformation` in the `getAnalysisResults()` function. As above, we start with an initial design, which in this situation is arbitrary and can be considered as a dummy design. Note that neither the number of stages nor the information rates need to be fixed.

```
# Dummy design
dummy <- getDesignGroupSequential(sided = 1, alpha = 0.025, typeOfDesign = "asOF")
```

The survival design was planned with a maximum of 387 events, the first interim took place after the observation of 205 events, the second after 285 events. Specifying the parameter `maxInformation` makes it now extremly easy to perform the analysis for the first and the second stage. Assume that we have observed log-rank statistics 1.87 and 2.19 at the first and the second interim, respectively. This observation together with the event numbers is defined in the `getDataset()` function through

```
dataSurvival <- getDataset(
    cumulativeEvents = c(205, 285),
    cumulativeLogRanks = c(1.87, 2.19)
)
```

Note that it is important to define **cumulative**Events and **cumulative**LogRanks because otherwise the stage wise events and logrank statistics should be entered (in the given case, these will be calculated).

We now can enter the planned maximum number of events in the `getAnalysisResults()` function as follows:

```
testResults <- getAnalysisResults(
    design = dummy,
    dataInput = dataSurvival,
    maxInformation = 387
)
```

This provides the summary:

Analysis results for a survival endpoint

Sequential analysis with 3 looks (group sequential design). The results were calculated using a two-sample logrank test (one-sided). H0: hazard ratio = 1 against H1: hazard ratio > 1.

Stage	1	2	3
Fixed weight	0.53	0.736	1
Efficacy boundary (z-value scale)	2.867	2.393	2.011
Cumulative alpha spent	0.0021	0.0090	0.0250
Stage level	0.0021	0.0084	0.0222
Cumulative effect size	1.299	1.296	
Overall test statistic	1.870	2.190	
Overall p-value	0.0307	0.0143	
Test action	continue	continue	
Conditional rejection probability	0.1927	0.3987	
95% repeated confidence interval	[0.870; 1.938]	[0.976; 1.721]	
Repeated p-value	0.1159	0.0380	

We see that the boundaries are correctly calculated according to the observed information rates. If there is overrunning, i.e., the final analysis was conducted after 393 rather than the planned 387 events, first define the observed dataset

```
dataSurvival <- getDataset(
    cumulativeEvents = c(205, 285, 393),
    cumulativeLogRanks = c(1.87, 2.19, 2.33)
)
```

and then use the `getAnalysisResults()` function as before:

```
testResults <- getAnalysisResults(
    design = dummy,
    dataInput = dataSurvival,
    maxInformation = 387
)
```

The messages describe the way of how the critical value for the last stage using the recalculated information rates (leaving the critical values for the first two stages unchanged) was calculated. This way was described in Section 5. The last warning indicates that for this case, since there is no "natural" family of decision boundaries, repeated p-values for the final stage of the trial are not calculated.

The summary shows that indeed the recalculated boundary for the last stage and the already used boundaries for the first two stages are used for decision making:

Analysis results for a survival endpoint

Sequential analysis with 3 looks (group sequential design). The results were calculated using a two-sample logrank test (one-sided). H0: hazard ratio = 1 against H1: hazard ratio > 1.

Stage	1	2	3
Fixed weight	0.522	0.725	1
Efficacy boundary (z-value scale)	2.867	2.393	2.014
Cumulative alpha spent	0.0021	0.0090	0.0250
Stage level	0.0021	0.0084	0.0220
Cumulative effect size	1.299	1.296	1.265
Overall test statistic	1.870	2.190	2.330
Overall p-value	0.0307	0.0143	0.0099
Test action	continue	continue	reject
Conditional rejection probability	0.1910	0.3883	
95% repeated confidence interval	[0.870; 1.938]	[0.976; 1.721]	[1.032; 1.550]
Repeated p-value	0.1159	0.0380	
Final p-value			0.0148
Final confidence interval			[1.023; 1.534]
Median unbiased estimate			1.255

We also can consider the case of underrunning which is the case if, for example, it was decided **before conducting the analysis** that, say, also if up to 3 less events than the considered maximum number will be observed, this should be considered as the final analysis (i.e., the final stage is reached if 384 or more events were observed). Inserting the parameter `informationEpsilon` in the `getAnalysisResults()` function can be used for this. There are two ways for defining this parameter. You can do it

1. in an absolute sense: the parameter `informationEpsilon` specifies the number of events to be allowed to deviate from the maximum number of events. This is achieved by specifying a positive integer number for `informationEpsilon`.
2. in a relative sense: if a number x < 1 for `informationEpsilon` is specified, the stage is considered as the final stage if x% of `maxInformation` is observed.

Both ways yield a correct calculation of the critical value to be used for the final stage. Suppose, for example, 385 events were observed and `informationEpsilon` was set equal to 3. Then, since `387 - 385 < 3`, this is an underrunning case and the critical value at the final stage is provided in the summary:

```
dataSurvival <- getDataset(
    cumulativeEvents = c(205, 285, 385),
    cumulativeLogRanks = c(1.87, 2.19, 2.21)
)
testResults <- getAnalysisResults(
    design = dummy,
    dataInput = dataSurvival,
    maxInformation = 387,
    informationEpsilon = 3
)
```

Analysis results for a survival endpoint

Sequential analysis with 3 looks (group sequential design). The results were calculated using a two-sample logrank test (one-sided). H0: hazard ratio = 1 against H1: hazard ratio > 1.

Stage	1	2	3
Fixed weight	0.532	0.74	1
Efficacy boundary (z-value scale)	2.867	2.393	2.010
Cumulative alpha spent	0.0021	0.0090	0.0250
Stage level	0.0021	0.0084	0.0222
Cumulative effect size	1.299	1.296	1.253
Overall test statistic	1.870	2.190	2.210
Overall p-value	0.0307	0.0143	0.0136
Test action	continue	continue	reject
Conditional rejection probability	0.1932	0.4023	
95% repeated confidence interval	[0.870; 1.938]	[0.976; 1.721]	[1.021; 1.538]
Repeated p-value	0.1159	0.0380	
Final p-value			0.0175
Final confidence interval			[1.016; 1.524]
Median unbiased estimate			1.246

We see that again the recalculated boundary for the last stage and the already used boundaries for the first two stages are used for decision making.

In summary, `maxInformation` in the `getAnalysisResults()` function can be used to perform an α-spending function approach in practice. Also, if at the analysis stage overrunning or (pre-defined) underrunnig takes place the use of the parameters `maxInformation` and `informationEpsilon` in the function provides an easy was to perform a correct analysis with the specified design.

System: rpact 3.3.2, R version 4.2.1 (2022-06-23 ucrt), platform: x86_64-w64-mingw32

To cite R in publications use:

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Um Paket 'rpact' in Publikationen zu zitieren, nutzen Sie bitte:

Wassmer G, Pahlke F (2022). *rpact: Confirmatory Adaptive Clinical Trial Design and Analysis*. https://www.rpact.org, https://www.rpact.com, https://github.com/rpact-com/rpact.

 file