Planning a Survival Trial with rpact

Gernot Wassmer and Friedrich Pahlke

Last change: 15 November, 2022

Contents

Summary

This R Markdown document provides an example for planning a trial with a survival endpoint using rpact thereby illustrating the different ways of entering recruitment schemes. It also demonstrates the use of the survival simulation function.

1 Types of calculations

First, load the rpact package

```
library(rpact)
packageVersion("rpact") # version should be version 2.0.5 or later
```

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

A survival trial is planned to be performed with one interim stage and using an O'Brien & Fleming type α-spending approach at $\alpha = 0.025$. The interim is planned to be performed after half of the necessary events were observed. It is assumed that the median survival time is 18 months in the treatment group, and 12 months in the control. We assume that the drop-out rate is 5% after 1 year and the drop-out time is exponentially distributed.

1.1 Accrual and follow-up time given

The patients should be recruited within 12 months assuming uniform accrual. We assume an additional follow-up time of 12 months, i.e., the study should be conducted within 2 years. We also assume the survival time to be exponentially distributed.

In this simplest example, accrual and follow-up time needs to be specified and the necessary number of events and patients (total and per month) in order to reach power 90% with the assumed median survival times will be calculated.

The effect size is defined in terms of `lambda1` and `lambda2` (you can also specify `lambda2` and `hazardRatio`) and the function `getLambdaByMedian()` is used (for `lambda2` the direct calculation is illustrated):

```
dGS <- getDesignGroupSequential(kMax = 2, typeOfDesign = "asOF", beta = 0.1)

x1 <- getSampleSizeSurvival(dGS,
    lambda1 = getLambdaByMedian(18), lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = 12, followUpTime = 12
)
kable(summary(x1))
```

Sample size calculation for a survival endpoint

Sequential analysis with a maximum of 2 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: treatment lambda(1) = 0.039, control lambda(2) = 0.058, accrual time = 12, accrual intensity = 38.9, dropout rate(1) = 0.05, dropout rate(2) = 0.05, dropout time = 12, power 90%.

Stage	1	2
Information rate	50%	100%
Efficacy boundary (z-value scale)	2.963	1.969
Overall power	0.2525	0.9000
Expected number of subjects	467.3	
Number of subjects	467.3	467.3
Cumulative number of events	128.3	256.5
Expected study duration	21.3	
Cumulative alpha spent	0.0015	0.0250
One-sided local significance level	0.0015	0.0245
Efficacy boundary (t)	0.593	0.782
Exit probability for efficacy (under H0)	0.0015	
Exit probability for efficacy (under H1)	0.2525	

Legend:

- *(t)*: treatment effect scale

We conclude that a maximum number of 257 events needs to be observed to reach 90% power. Under the assumed accrual and follow-up time (and the median survival times) this number of events are expected to be observed if 468 subjects are observed at the end of the trial. This corresponds to about 39 (= `accrualIntensity`) patient per month to be accrued.

The interim analysis is performed after 128.3 events (which in practice needs to be rounded up or down). This number is expected to be observed at analysis time 13.14.

Note that in this case you can also enter the accrual time as `accrualTime = c(0,12)` and so `accrualTime = 12` is just a shortcut (see below).

1.2 Accrual time and maximum number of patients given

Assume that accrual stops after 16 months with 25 patients per month, i.e., after 400 patients were recruited. We now can estimate the necessary follow-up time such that 257 events will be observed at the end of the trial.

There are two ways of defining the situation of *given maximum number of subjects*: You can either specify `accrualTime` and `maxNumberOfSubjects` directly; or you specify `accrualTime` and `accrualIntensity`.

In both cases, at given accrual time and number of patients, the follow-up time is calculated.

This is the code and the output for the first case:

```
x2 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = c(0, 16), accrualIntensity = 25
)

ceiling(x2$maxNumberOfSubjects)
```

```
## [1] 400
```

```
x2$followUpTime
```

```
## [1] 15.96226
```

```
x2$analysisTime
```

```
##          [,1]
## [1,] 16.82864
## [2,] 31.96226
```

This is the code and the output for the second case:

```
x2 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = c(0, 16), maxNumberOfSubjects = 400
)

x2$accrualIntensity
```

```
## [1] 25
```

```
x2$followUpTime
```

```
## [1] 15.96226
```

```
x2$analysisTime
```

```
##          [,1]
## [1,] 16.82864
## [2,] 31.96226
```

Note that for these two cases entering the accrual time in the form of `c(0,16)` is mandatory, i.e., you cannot enter just the end of accrual which was the shortcut before.

Alternatively, you might enter the `accrualTime` as a list in the form

```
atList <- list(
    "0 - <16" = 25
)
```

and then enter `accrualTime = atList` in `getSampleSizeSurvival()`:

```
x2 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = atList
)

x2$accrualIntensity
```

```
## [1] 25
```

```
x2$followUpTime
```

```
## [1] 15.96226
```

```
x2$analysisTime
```

```
##          [,1]
## [1,] 16.82864
## [2,] 31.96226
```

1.3 Follow-up time and absolute intensity given

Assume now that 25 patients can be recruited each month and that there is uniform accrual. We now want to estimate the necessary accrual time if the *planned follow-up time is given* and equal to 12 as before.

Here the end of accrual and the number of patients is calculated at given follow-up time and absolute accrual intensity:

```
x3 <- getSampleSizeSurvival(dGS,
    hazardRatio = 2 / 3, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = 0, accrualIntensity = 25, followUpTime = 12
)

ceiling(x3$maxNumberOfSubjects)
```

```
## [1] 435
```

```
x3$accrualTime
```

```
## [1] 17.38334
```

```
x3$analysisTime
```

```
##          [,1]
## [1,] 16.79806
## [2,] 29.38334
```

1.4 Staggered patient entry

This case illustrates how the accrual of subjects can be entered if there is piecewise accrual over predefined intervals where it is assumed that accrual within the intervals is uniform.

For example, assume that in the first 3 months 15 patients, in the second 3 months 20 patients, and after 6 months 25 patients per month can be accrued.

If the follow-up time at given number of patients is to be calculated (Section 1.2), it is simply to enter `accrualTime = c(0, 3, 6, 16)` and `accrualIntensity = c(15, 20, 25)` as follows (`maxNumberOfSubjects` will be calculated):

```
x4 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = c(0, 3, 6, 16), accrualIntensity = c(15, 20, 25)
)

ceiling(x4$maxNumberOfSubjects)
```

```
## [1] 355
```

```
x4$followUpTime
```

```
## [1] 23.60973
```

```
x4$analysisTime
```

```
##          [,1]
## [1,] 18.83368
## [2,] 39.60973
```

Alternatively, you can enter

```
atList <- list(
    "0 - < 3" = 15,
    "3 - < 6" = 20,
    "6 - <= 16" = 25
)
```

and run

```
x4 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = atList
)

ceiling(x4$maxNumberOfSubjects)
x4$followUpTime
x4$analysisTime
```

If `maxNumberOfSubjects` is specified, the corresponding list definition is as follows:

```
atList <- list(
    "0 - < 3" = 15,
    "3 - < 6" = 20,
    "6 - Inf" = 25
)
```

and

```
x4 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = atList, maxNumberOfSubjects = 355
)

ceiling(x4$accrualTime)
x4$followUpTime
x4$analysisTime
```

does the job.

If the follow-up time is given and the end of accrual is to be calculated (Section 1.3), this can be achieved as follows:

```
x5 <- getSampleSizeSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    accrualTime = c(0, 3, 6), accrualIntensity = c(15, 20, 25), followUpTime = 12
)

ceiling(x5$maxNumberOfSubjects)
```

```
## [1] 421
```

```
x5$accrualTime
```

```
## [1]  3.00000  6.00000 18.61383
```

```
x5$analysisTime
```

```
##          [,1]
## [1,] 18.28469
## [2,] 30.61383
```

2 Verify results by simulation

Assume that the study is planned with 257 events and 400 patients under the assumptions that accrual stops after 16 months with 25 patients per month and the staggered patient entry from above. We now want to verify by simulation the correctness of the results obtained by the analytical formulae.

For this, we first use the function `getPowerSurvival()` in order to obtain the test characteristics if the study is performed with 257 events. The analysis time is obtained by the analytical formulae and we verify that the power is indeed exceeding 90% (note that we have to specify `directionupper = FALSE` in order to indicate that p-values are getting small for *negative* values of the test statistics, i.e., for observed hazard ratios smaller than 1):

```
y3 <- getPowerSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    accrualTime = c(0, 3, 6, 16), accrualIntensity = c(15, 20, 25),
    maxNumberOfEvents = 257, directionUpper = FALSE
)
y3$analysisTime
```

```
##          [,1]
## [1,] 18.85699
## [2,] 39.74904
```

```
y3$overallReject
```

```
## [1] 0.9005251
```

Practically the same result is obtained with the simulation function `getSimulationSurvival()` (note that `plannedEvents = c(129,257)` needs to be specified in the simulation tool instead of `maxNumberOfEvents = 257` in the power calculation:

```
z3 <- getSimulationSurvival(dGS,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    dropoutRate1 = 0.05, dropoutRate2 = 0.05, dropoutTime = 12,
    maxNumberOfIterations = 10000,
    accrualTime = c(0, 3, 6, 16), accrualIntensity = c(15, 20, 25),
    plannedEvents = c(129, 257), directionUpper = FALSE, seed = 12345
)
z3$analysisTime
```

```
##          [,1]
## [1,] 18.91939
## [2,] 39.63640
```

```
z3$overallReject
```

```
## [1] 0.9054
```

3 Assess sample size reassessment in adaptive survival design

Assume now that a sample size increase up to a ten-fold of the originally planned number of events is foreseen. Conditional power 90% *based on the observed hazard ratios* is used to increase the number of events. We can now assess by simulation the magnitude of power increase when using the appropriate method.

Note that we have to make sure that enough subjects are used in the simulation. For this, we set `maxNumberOfSubjects = 2000` and no drop-outs.

Since we perform a data driven sample size recalculation (SSR), we define an inverse normal design with the same parameters as the original group sequential design:

```
dIN <- getDesignInverseNormal(kMax = 2, typeOfDesign = "asOF", beta = 0.1)
```

The following code simulates the SSR with conditional power 90% under the observed hazard ratio. `minNumberOfEventsPerStage = c(NA,128)`, and `maxNumberOfEventsPerStage = 10*c(NA,128)` defines:

- no sample size decrease
- up to a 10 fold increase of number of events for the final stage.

This is compared with the situation without SSR:

```
#  Without SSR
z0 <- getPowerSurvival(dIN,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    accrualTime = c(0, 16), maxNumberOfSubjects = 2000, maxNumberOfEvents = 257,
    directionUpper = FALSE
)
z0$analysisTime
```

```
##           [,1]
## [1,]  6.911003
## [2,] 10.010260
```

```
z0$overallReject
```

```
## [1] 0.9005251
```

```
#  With SSR
z4 <- getSimulationSurvival(dIN,
    lambda1 = log(2) / 18, lambda2 = log(2) / 12,
    maxNumberOfIterations = 10000,
    accrualTime = c(0, 16), maxNumberOfSubjects = 2000, plannedEvents = c(129, 257),
    directionUpper = FALSE, conditionalPower = 0.9,
    minNumberOfEventsPerStage = c(NA, 128),
    maxNumberOfEventsPerStage = 10 * c(NA, 128), seed = 23456
)
z4$analysisTime
```

```
##              [,1]
## [1,]  6.914569
## [2,] 15.037551
```

```
z4$overallReject
```

```
## [1] 0.9842
```

Finally, we simulate the Type I error rate when using

- the group sequential method
- the inverse normal method

for analysing the data. The difference here is that the group sequential design uses the logrank tests for the test decision whereas with the combination test the *increments* of the sequence of logrank statistics are combined with the use of the inverse normal combination function. This assures Type I error control also the data-driven SSR.

The following simulation compares the Type I error rate of the inverse normal method with the Type I error rate of the (inappropriate) use of the group sequential method:

```
dGS <- getDesignGroupSequential(kMax = 2, typeOfDesign = "asOF")
dIN <- getDesignInverseNormal(kMax = 2, typeOfDesign = "asOF")

IN <- getSimulationSurvival(dIN,
    hazardRatio = 1,
    maxNumberOfIterations = 10000,
    accrualTime = c(0, 16), maxNumberOfSubjects = 2000, plannedEvents = c(129, 257),
    directionUpper = FALSE, conditionalPower = 0.9,
    minNumberOfEventsPerStage = c(NA, 128),
    maxNumberOfEventsPerStage = 10 * c(NA, 128), seed = 34567
)

GS <- getSimulationSurvival(dGS,
    hazardRatio = 1,
    maxNumberOfIterations = 10000,
    accrualTime = c(0, 16), maxNumberOfSubjects = 2000, plannedEvents = c(129, 257),
    directionUpper = FALSE, conditionalPower = 0.9, minNumberOfEventsPerStage = c(NA, 128),
    maxNumberOfEventsPerStage = 10 * c(NA, 128), seed = 45678
)

IN$overallReject
```

```
## [1] 0.0255
```

```
GS$overallReject
```

```
## [1] 0.0352
```

We see that indeed the Type I error rate is controlled for the inverse normal but not for the group sequential method (see also this vignette).

4 References

1. Gernot Wassmer and Werner Brannath, *Group Sequential and Confirmatory Adaptive Designs in Clinical Trials*, Springer 2016, ISBN 978-3319325606

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