Supplementing and enhancing rpact's graphical capabilities with ggplot2

Yannic Schmidt, Stefan Englert, Martina Kron, Gernot Wassmer, and Friedrich Pahlke

12 Dezember, 2019

Contents

Summary **1**

1 Introduction **1**

2 Initialize the group-sequential and adaptive inverse-normal design **2**

3 Modification of the rpact base plots with ggplot2 (aesthetic) commands **4**

4 Illustrating Type I error control of adaptive designs with rpact and ggplot2 **8**
4.1 Simmulating the Type I error rate 8
4.2 Extraction of the simulated Type I error rate of both designs 11
4.3 Final Plot 11

5 Summary **13**

6 References **13**

7 Disclaimer **13**

Summary

The aim of this R Markdown document is to give a brief description on **how easy it is to supplement and enhance plots** generated in rpact by use of the ggplot2 package and associated language.

1 Introduction

We will illustrate the generation of the plots by assessing the simulated Type I error rate of a group-sequential and adaptive inverse-normal design in a sample size recalculation setting. It is well known that the adaptive inverse-normal designs in contrast to group-sequential designs controls the overall Type I error rate even in a setting of sample size recalculation. The goal of this exercise is to present this fact in a single comprehensive output.

With *rpact* it is very convenient to calculate the necessary components: We will analyze both design choices by *rpact*, retrieve the results and arrange them by the means of *ggplot2*.

Throughout this document, we will use the default parameter settings from *rpact* which assume a one-sided significance level of 0.025.

Loading the packages

We start by loading the required two packages.

```
library(rpact)
packageVersion("rpact")
```

```
## [1] '2.0.6'
```

```
library(ggplot2)
```

2 Initialize the group-sequential and adaptive inverse-normal design

After the two packages were loaded, the group-sequential (dGS) and adaptive inverse-normal (dIN) design can be generated.

```
dGS <- getDesignGroupSequential()
dIN <- getDesignInverseNormal()
```

We can now take a look into both designs and inspect their default values.

```
dGS
```

```
## Design parameters and output of group sequential design:
##
## User defined parameters: not available
##
## Derived from user defined parameters: not available
##
## Default parameters:
##   Type of design                          : OF
##   Maximum number of stages                : 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                               : 0.00000001
##
## 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
##
## ISSUES (parameters with undefined type):
##   Stages                                  :
```

```
dIN
```

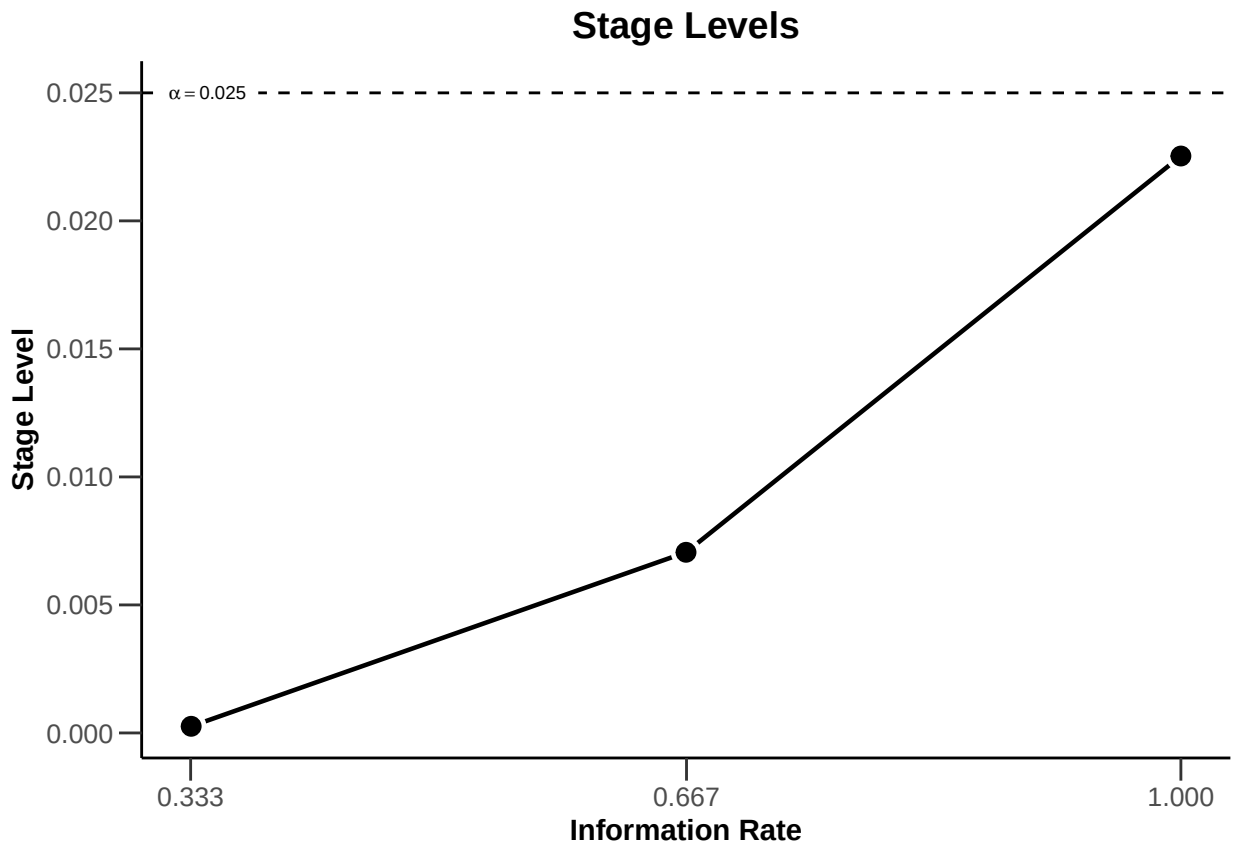
```
## Design parameters and output of inverse normal design:
##
## User defined parameters:
##   Two-sided power                         : FALSE
##
## Derived from user defined parameters: not available
##
## Default parameters:
##   Type of design                          : OF
##   Maximum number of stages                : 3
##   Information rates                       : 0.333, 0.667, 1.000
##   Significance level                      : 0.0250
##   Type II error rate                      : 0.2
```

```
##   Test                                              : one-sided
##   Tolerance                                         : 0.00000001
##
## 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
##
## ISSUES (parameters with undefined type):
##   Stages                                            :
```

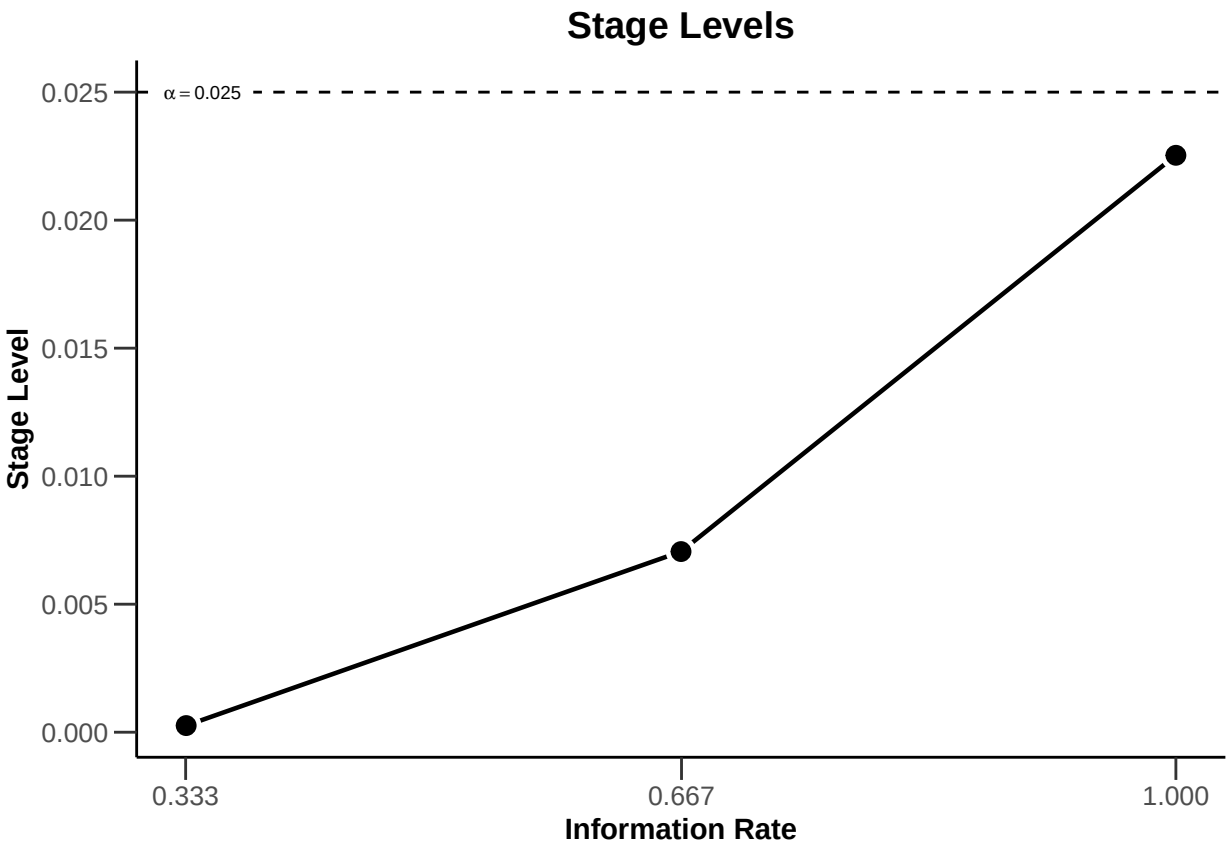
In the output we see some important characteristics like the type of design (OF=O'Brien and Fleming), the maximum number of stages (3), the information rates (0.333, 0.667, 1.000), the significance level (0.025) and the stage level information (0.0002592, 0.0070554, 0.0225331).

We can also easily plot the stage level information by means of *rpact*.

```
plot(dGS, type = 3)
```




```
plot(dIN, type = 3)
```

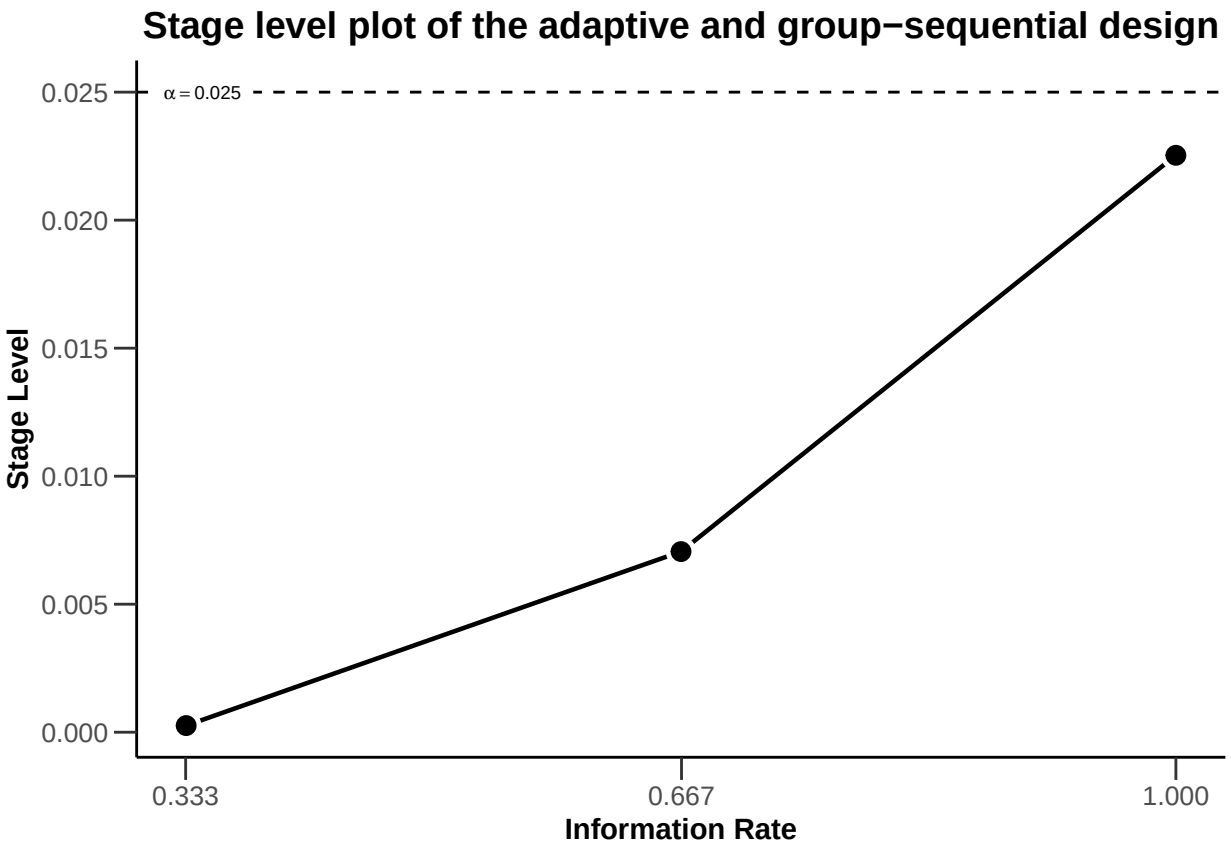


With these commands, *rpact* creates two plots that show the three stage levels (0.0025, 0.0070554, 0.0225331) visually together with the corresponding information rates (0.333, 0.667, 1.000) and the significance level (0.025, dashed line).

3 Modification of the rpact base plots with ggplot2 (aesthetic) commands

We note that the two prior plots are identical. This is expected as the same boundary type (O'Brien and Fleming) was used for both designs. So, we might want to restrict ourselves to present only one stage level plot. Instead, we can include this information in a custom title. This is achieved by combining the *rpact* "base" plot with *ggplot2* aesthetics commands as follows:

```
plot(dGS, type = 3) +
  ggtitle("Stage level plot of the adaptive and group-sequential design")
```

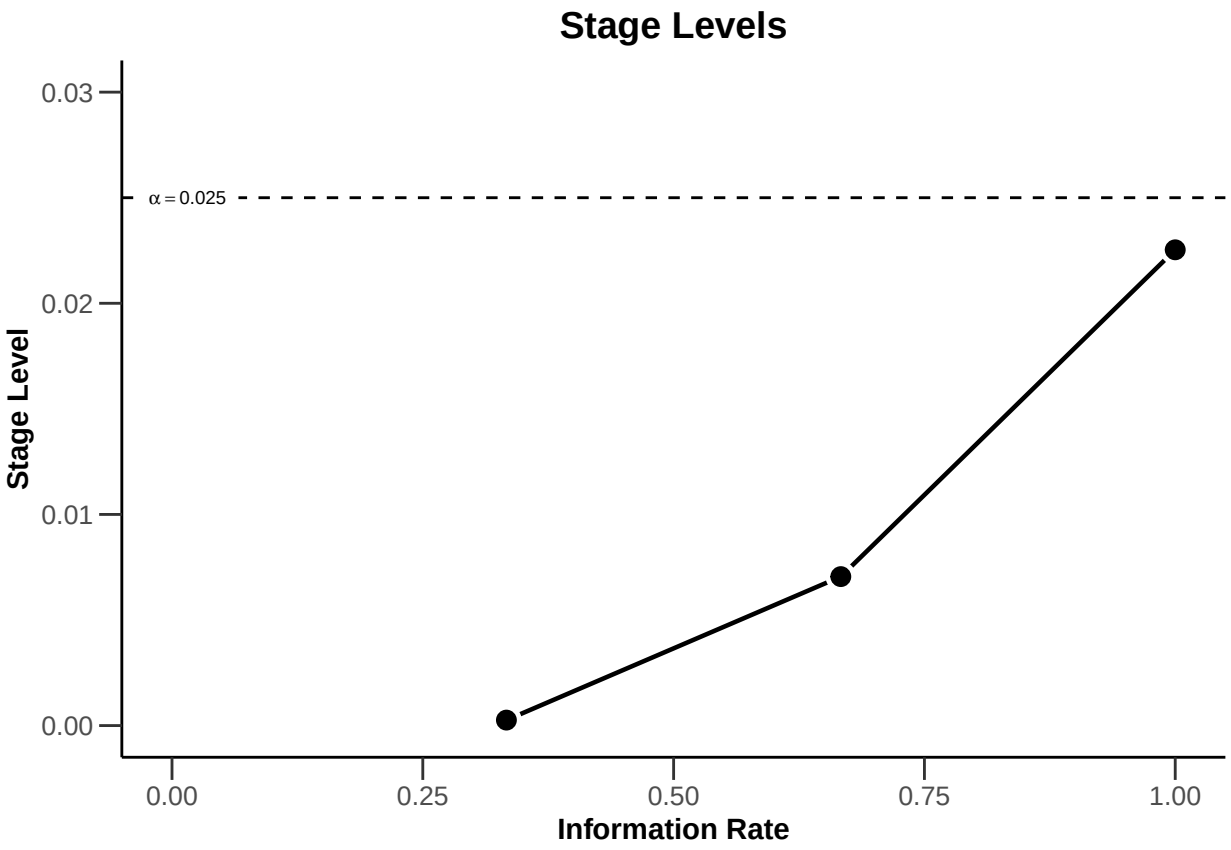


All *ggplot2* aesthetic commands which modify an existing plot are initiated by a plus sign "+". Thereafter, we can add the particular command in the *ggplot2* language to specify how we want to modify the plot. Luckily, almost all *rpact* graphics are compatible with *ggplot2*.

Modifications are not restriction to additions to existing plots, as demonstrated by adding a new title, but they can also re-model the initial plot. For example, we can change the limits of the x- and y-axis post-hoc.

```
plot(dGS, type = 3) + xlim(0, 1) + ylim(0, dGS$alpha + 0.005)
```

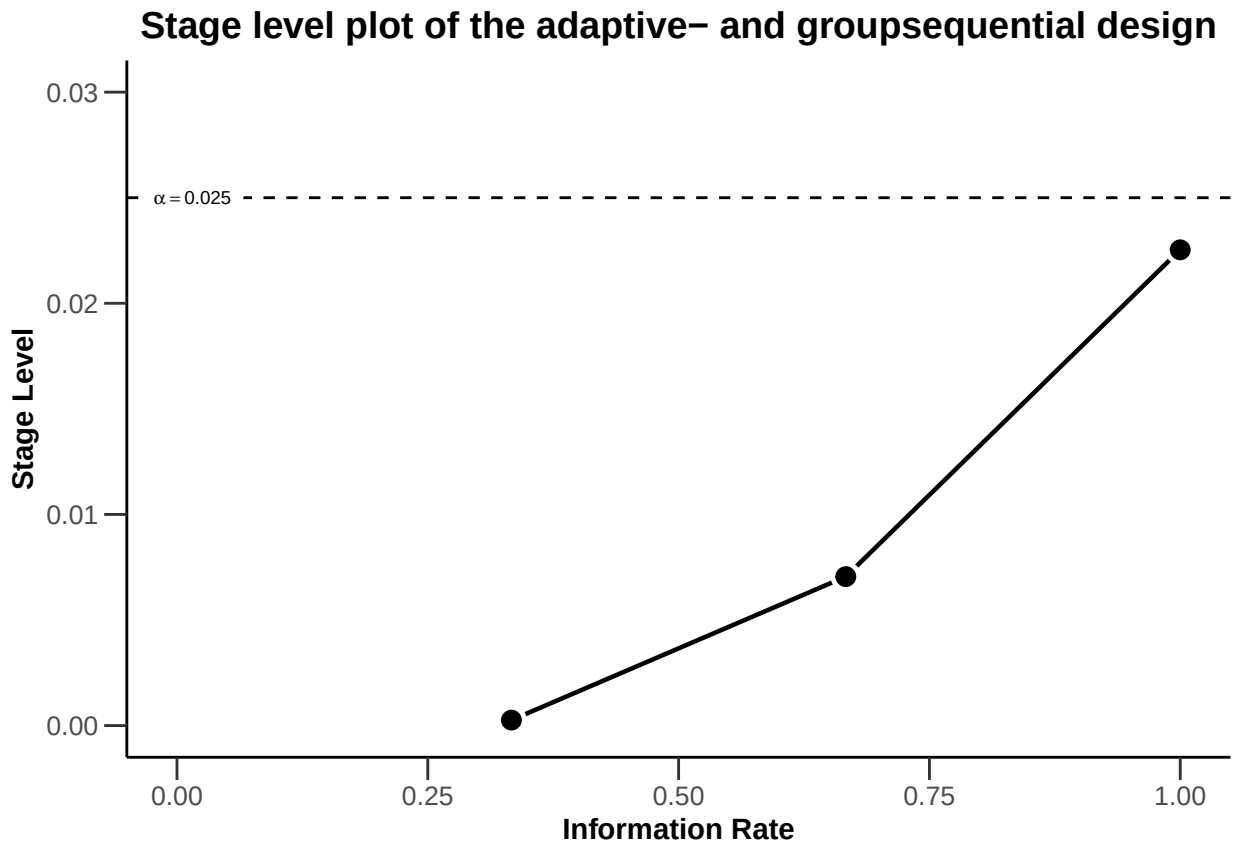
```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Or do both:

```
plot(dGS, type = 3) +
  ggtitle("Stage level plot of the adaptive- and groupsequential design") +
  xlim(0, 1) + ylim(0, dGS$alpha + 0.005)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

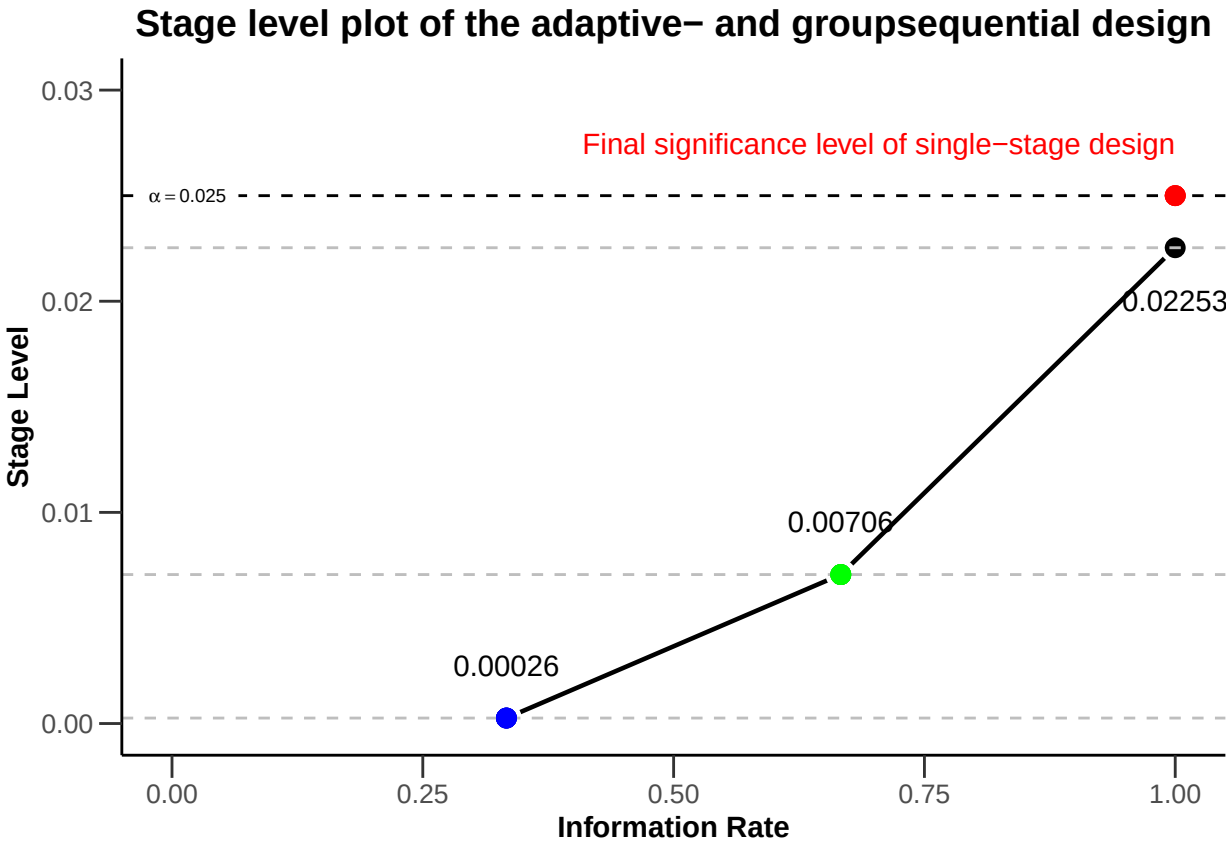


To make the plot more self-explanatory, we might want to further add

- a visual highlight for the significance level of a single stage design by using the `geom_point` and `annotate` commands
- a better visual illustration of the local stage levels by adding horizontal reference lines by usign `geom_hline` commands
- color to the three different stages in the plot by using the aesthetic command `geom_point`
- the numerical values of the stage levels to plot by using appropriate `annotate` commands

```
plot(dGS, type = 3) +
  ggtitle("Stage level plot of the adaptive- and groupsequential design") +
  xlim(0, 1) + ylim(0, dGS$alpha + 0.005) +
  geom_point(aes(x = 1, y = dGS$alpha), colour = "red", size = 3) +
  annotate(geom = "text", label = "Final significance level of single-stage design",
    color = "red", x = dGS$informationRates[3], y = dGS$alpha, vjust = -2, hjust = 1) +
  geom_hline(yintercept = dGS$stageLevels,linetype = "dashed", color = "grey") +
  geom_point(aes_(x = dGS$informationRates[1], y = dGS$stageLevels[1]),
    color = "blue", size = 3) +
  geom_point(aes_(x = dGS$informationRates[2], y = dGS$stageLevels[2]),
    color = "green", size = 3) +
  annotate(geom = "text", label = round(dGS$stageLevels[1],5),
    x = dGS$informationRates[1], y = dGS$stageLevels[1], vjust = -2) +
  annotate(geom = "text", label = round(dGS$stageLevels[2],5),
    x = dGS$informationRates[2], y = dGS$stageLevels[2], vjust = -2) +
  annotate(geom = "text", label = round(dGS$stageLevels[3],5),
    x = dGS$informationRates[3], y = dGS$stageLevels[3], vjust = 3)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



Note that the commands pull the required information directly from the design object and thus would automatically adjust to design changes like, for example, a switch to Pocock boundaries.

4 Illustrating Type I error control of adaptive designs with rpact and ggplot2

In the last part of this document we investigate the Type I error rate of the chosen group sequential and adaptive inverse normal design in a sample size recalculation setting and present the simulation results visual in the plot.

4.1 Simmulating the Type I error rate

Using `getSimulationMeans()` we can easily simulate the operating characteristics of designs. To simulate the Type I error rate, we specify the null hypothesis as an alternative hypothesis with zero therapy effect (`alternative = 0`). The simulated overall rejection rate of the designs then corresponds to the Type I error rate.

We assume a scenario with two interim analyses after 20 and 40 subjects and a final analysis after 60 subjects. We allow for a recalculation of the sample size for the next stage based on a conditional power of 80%. The recalculated sample size per stage is limited to be between 20 and 80. We perform a total of 100'000 simulations to get sufficient precision for the simulated Type I error rate. The seed = 12345 ensures that the simulated values are reproduced, e.g., for documentary purposes.

```
nsim <- 100000
seed <- 12345
dGSsim <- getSimulationMeans(dGS,
  plannedSubjects = c(20,40,60),
  conditionalPower = 0.8, alternative = 0,
  minNumberOfSubjectsPerStage = c(20,20,20),
  maxNumberOfSubjectsPerStage = c(20,80,80),
  maxNumberOfIterations = nsim,
  seed = 12345)
dGSsim
```

```
## Simulation of means (group sequential design):
##
## User defined parameters:
##   Alternatives                                : 0
##   Planned subjects                            : 20, 40, 60
##   Minimum number of subjects per stage        : 20, 20, 20
##   Maximum number of subjects per stage        : 20, 80, 80
##   Conditional power                           : 0.8
##   Maximum number of iterations                : 100000
##   Seed                                        : 12345
##
## Default parameters:
##   Standard deviation                          : 1
##   Treatment groups                            : 2
##   Planned allocation ratio                    : 1
##   Direction upper                             : TRUE
##   Theta H0                                    : 0
##   Mean ratio                                  : FALSE
##
## Results:
##   Effect                                      : 0
##   Iterations [1]                              : 100000
##   Iterations [2]                              : 99976
##   Iterations [3]                              : 99104
##   Sample sizes [1]                            : 20
##   Sample sizes [2]                            : 77.176
##   Sample sizes [3]                            : 78.686
##   Reject per stage [1]                        : 0.00024
##   Reject per stage [2]                        : 0.00872
##   Reject per stage [3]                        : 0.02297
##   Overall reject                              : 0.0319
##   Futility stop per stage [1]                 : 0
##   Futility stop per stage [2]                 : 0
##   Futility stop                               : 0
##   Early stop                                  : 0.00896
##   Expected number of subjects                 : 175.1
##   Cond. power (achieved) [1]                  : NA
##   Cond. power (achieved) [2]                  : 0.1282
##   Cond. power (achieved) [3]                  : 0.0848
##
## Simulated data:
##   Number of subjects [1]                      : median [range]: 20 [20 - 20]; mean +/-sd: 20 +/-0
##   Number of subjects [2]                      : median [range]: 80 [20 - 80]; mean +/-sd: 77.176 +/
```

```
##   Number of subjects [3]                          : median [range]: 80 [20 - 80]; mean +/-sd: 78.686 +/
##   Test statistic [1]                              : median [range]: 0.006 [-4.563 - 4.33]; mean +/-sd: (
##   Test statistic [2]                              : median [range]: 0.02 [-4.21 - 4.168]; mean +/-sd: 0
##   Test statistic [3]                              : median [range]: -0.001 [-4.797 - 3.956]; mean +/-sd
##   Cond. power (achieved) [2]                      : median [range]: 0 [0 - 1]; mean +/-sd: 0.128 +/-0.2
##   Cond. power (achieved) [3]                      : median [range]: 0 [0 - 0.959]; mean +/-sd: 0.085 +/
##
## Legend:
##   [k]: values at stage k
```

```
dINsim <- getSimulationMeans(dIN,
  plannedSubjects = c(20,40,60),
  conditionalPower = 0.8, alternative = 0,
  minNumberOfSubjectsPerStage = c(20,20,20),
  maxNumberOfSubjectsPerStage = c(20,80,80),
  maxNumberOfIterations = nsim,
  seed = 12345)
dINsim
```

```
## Simulation of means (inverse normal design):
##
## User defined parameters:
##   Alternatives                                  : 0
##   Planned subjects                              : 20, 40, 60
##   Minimum number of subjects per stage          : 20, 20, 20
##   Maximum number of subjects per stage          : 20, 80, 80
##   Conditional power                             : 0.8
##   Maximum number of iterations                  : 100000
##   Seed                                          : 12345
##
## Default parameters:
##   Standard deviation                            : 1
##   Treatment groups                              : 2
##   Planned allocation ratio                      : 1
##   Direction upper                               : TRUE
##   Theta H0                                      : 0
##   Mean ratio                                    : FALSE
##
## Results:
##   Effect                                        : 0
##   Iterations [1]                                : 100000
##   Iterations [2]                                : 99975
##   Iterations [3]                                : 99300
##   Sample sizes [1]                              : 20
##   Sample sizes [2]                              : 77.142
##   Sample sizes [3]                              : 78.748
##   Reject per stage [1]                          : 0.00025
##   Reject per stage [2]                          : 0.00675
##   Reject per stage [3]                          : 0.01807
##   Overall reject                                : 0.0251
##   Futility stop per stage [1]                   : 0
##   Futility stop per stage [2]                   : 0
##   Futility stop                                 : 0
##   Early stop                                    : 0.007
##   Expected number of subjects                   : 175.3
```

```
##   Cond. power (achieved) [1]                    : NA
##   Cond. power (achieved) [2]                    : 0.1280
##   Cond. power (achieved) [3]                    : 0.0806
##
## Simulated data:
##   Number of subjects [1]                        : median [range]: 20 [20 - 20]; mean +/-sd: 20 +/-0
##   Number of subjects [2]                        : median [range]: 80 [20 - 80]; mean +/-sd: 77.142 +/
##   Number of subjects [3]                        : median [range]: 80 [20 - 80]; mean +/-sd: 78.748 +/
##   Test statistic [1]                            : median [range]: 0.001 [-4.563 - 4.557]; mean +/-sd:
##   Test statistic [2]                            : median [range]: 0.002 [-4.839 - 4.129]; mean +/-sd:
##   Test statistic [3]                            : median [range]: -0.008 [-4.901 - 4.152]; mean +/-sd
##   Cond. power (achieved) [2]                    : median [range]: 0 [0 - 1]; mean +/-sd: 0.128 +/-0.2
##   Cond. power (achieved) [3]                    : median [range]: 0 [0 - 0.959]; mean +/-sd: 0.081 +/
##
## Legend:
##   [k]: values at stage k
```

4.2 Extraction of the simulated Type I error rate of both designs

Note that we can easily extract the simulated Type I error rate or overall rejection rates under the null hypothesis. This value is stored in the `overallReject` parameter of the simulation outputs.

```
dGSalpha <- dGSsim$overallReject
dGSalpha
```

```
## [1] 0.03193
```

```
dINalpha <- dINsim$overallReject
dINalpha
```

```
## [1] 0.02507
```

The results of the simulation show that, in a setting of sample size recalculation, the group-sequential design does not control the nominal significance level. The overall rejection rate of the group-sequential design (0.03193) is bigger than the significance level (0.025). The inverse normal design controls the Type I error rate as the overall rejection rate (0.02507) is close to the significance level of 0.025.

4.3 Final Plot

In the final plot of this example, we try to present the core learning of the previous section in a single overarching plot. The final plot should be *management ready*, meaning include all relevant information, be visually appealing and self-explanatory. The intention of the plot is to transfer the information that a group sequential design and adaptive inverse normal design with same parameters will feature the same stage levels at the corresponding information rates, the same overall significance level, but different overall rejection rates in a setting of sample size recalculation.

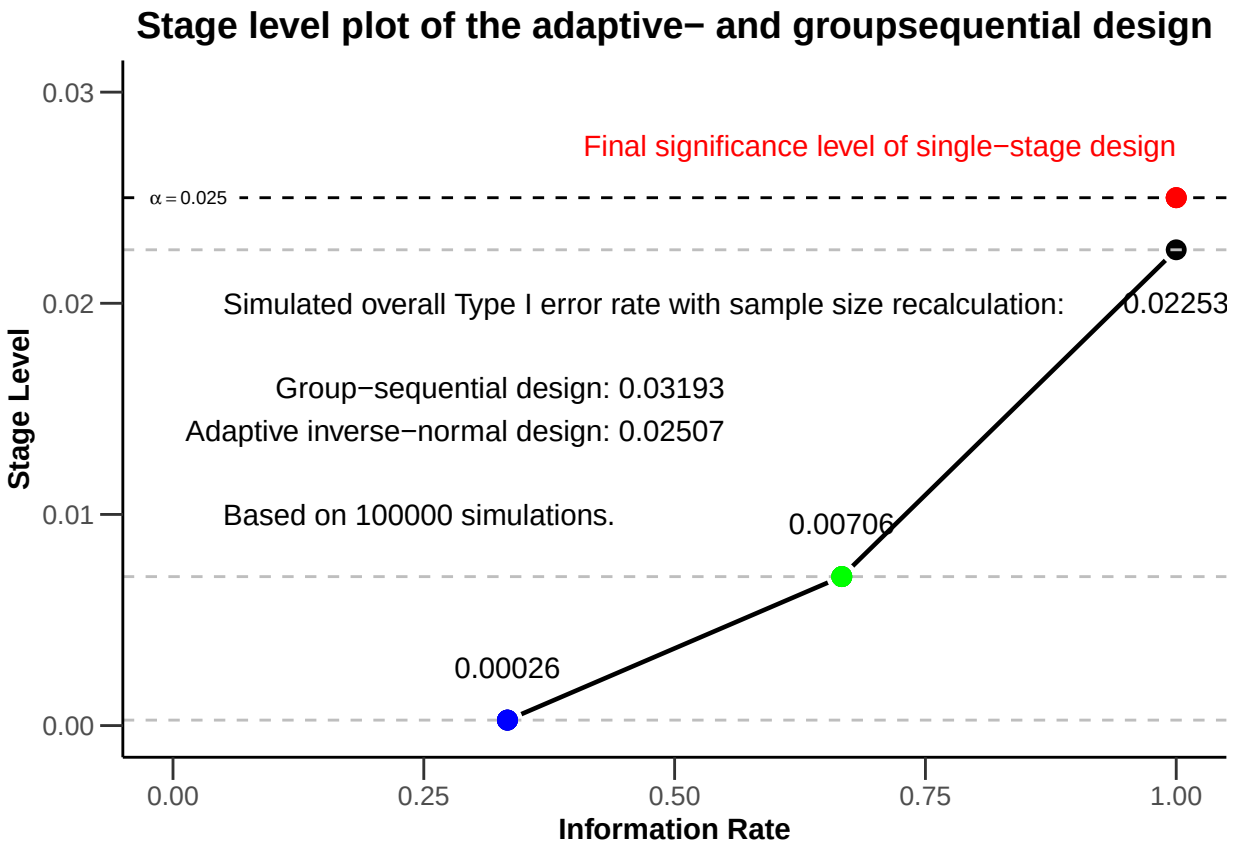
For the final plot, we start with all (plot and aesthetic) commands, which we used in the chapter *Modification of the rpact base chart with ggplot2 (aesthetic) commands*. Additional information from the simulation results was subsequently added in.

```
# create base ggplot object
g <- plot(dGS, type = 3)

# modify the rpact base plot as described above
g <- g + ggtitle("Stage level plot of the adaptive- and groupsequential design") +
  xlim(0, 1) + ylim(0, dGS$alpha + 0.005) +
  geom_point(aes(x = 1, y = dGS$alpha), colour = "red", size = 3) +
```

```
  annotate(geom = "text", label = "Final significance level of single-stage design",
    color = "red", x = dGS$informationRates[3], y = dGS$alpha, vjust = -2, hjust = 1) +
  geom_hline(yintercept = dGS$stageLevels,linetype = "dashed", color = "grey") +
  geom_point(aes_(x = dGS$informationRates[1], y = dGS$stageLevels[1]),
    color = "blue", size = 3) +
  geom_point(aes_(x = dGS$informationRates[2], y = dGS$stageLevels[2]),
    color = "green", size = 3)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

```
# add additional annotations
g <- g + annotate(geom = "text", label = round(dGS$stageLevels[1],5),
    x = dGS$informationRates[1], y = dGS$stageLevels[1], vjust = -2) +
  annotate(geom = "text", label = round(dGS$stageLevels[2],5),
    x = dGS$informationRates[2], y = dGS$stageLevels[2], vjust = -2) +
  annotate(geom = "text", label = round(dGS$stageLevels[3],5),
    x = dGS$informationRates[3], y = dGS$stageLevels[3], vjust = 3) +
  annotate("text", x = 0.55, y = 0.016, label =
    paste("Group-sequential design:", dGSalpha), hjust = 1) +
  annotate("text", x = 0.55, y = 0.014, label =
    paste("Adaptive inverse-normal design:", dINalpha), hjust = 1) +
  annotate("text", x = 0.05, y = 0.010, label =
    paste("Based on", nsim, "simulations."), hjust = 0) +
  annotate("text", x = 0.05, y = 0.020, label =
    "Simulated overall Type I error rate with sample size recalculation:", hjust = 0)

# generate and show plot
g
```



5 Summary

It was shown that *rpact* "base" charts can be modified with standard *ggplot2* commands. The implementation is very easy and intuitive as it follows the normal *ggplot2* syntax. The authors hope that this information is helpful for everyone who wants to further modify graphical outputs from *rpact*.

6 References

1. Gernot Wassmer and Werner Brannath, *Group Sequential and Confirmatory Adaptive Designs in Clinical Trials*, Springer 2016, ISBN 978-3319325606
2. R-Studio, *Data Visualization with ggplot2 - Cheat Sheet*, version 2.1, 2016, https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf

7 Disclaimer

Yannic Schmidt, Stefan Englert and Martina Kron are employees of AbbVie Inc. Gernot Wassmer and Friedrich Pahlke are employees of RPACT (R Programming for Adaptive Clinical Trials). All opinions and information in this presentation are from the authors and do not necessarily reflect the views of their employers.

System: rpact 2.0.6, R version 3.6.1 (2019-07-05), platform: x86_64-w64-mingw32

To cite R in publications use:

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

To cite package 'rpact' in publications use:

Gernot Wassmer and Friedrich Pahlke (2019). rpact: Confirmatory Adaptive Clinical Trial Design and Analysis. R package version 2.0.6. https://www.rpact.org

License