Validation Report for **adoptr** package

Kevin Kunzmann & Maximilian Pilz

2019-07-04

Contents

1 Introduction **5**
- 1.1 Preliminaries . . . 5
- 1.2 Scope . . . 5
- 1.3 Validation Scenarios . . . 7
- 1.4 Technical Setup . . . 13

2 Scenario I: large effect, point prior **15**
- 2.1 Details . . . 15
- 2.2 Variant I-1: Minimizing Expected Sample Size under Point Prior 16
- 2.3 Variant I-2: Minimizing Expected Sample Size under Null Hypothesis . . . 19
- 2.4 Variant I-3: Conditional Power Constraint . . . 22
- 2.5 Plot Two-Stage Designs . . . 25

3 Scenario II: Large effect, Gaussian prior **27**
- 3.1 Details . . . 27
- 3.2 Variant II-1: Minimizing Expected Sample Size under Point Prior 28
- 3.3 Variant II-2: Minimizing Expected Sample Size under Null Hypothesis . . . 30
- 3.4 Variant II-3: Conditional Power Constraint . . . 32
- 3.5 Plot Two-Stage Designs . . . 35

4 Scenario III: large effect, uniform prior **37**
- 4.1 Details . . . 37
- 4.2 Variant III.1: Convergence under prior concentration . . . 38

5 Scenario IV: smaller effect, point prior **43**
- 5.1 Details . . . 43
- 5.2 Variant IV-1: Minimizing Expected Sample Size under Point Prior 44
- 5.3 Variant IV-2: Increase Power . . . 47
- 5.4 Variant IV-3: Increase Type One Error rate . . . 51
- 5.5 Plot Two-Stage Designs . . . 55

6 Scenario V: Single-arm design, medium effect size **57**

6.1 Details 57
6.2 Variant V-1, sensitivity to integration order 58
6.3 Variant V-2, utility maximization 60
6.4 Variant V-3, n1-penalty 63
6.5 Variant V-4, n2-penalty 65

Chapter 1

Introduction

1.1 Preliminaries

R package validation for regulatory environments can be a tedious endeavour. The authors firmly believe that under the current regulation, there is no such thing as a 'validated R package': validation is by definition a process conducted by the *user*. This validation report merely aims at facilitating validation of **adoptr** as much as possible. No warranty whatsoever as to the correctness of **adoptr** nor the completeness of the validation report are given by the authors.

We assume that the reader is familiar with the notation an theoretical background of **adoptr**. Otherwise, the following resources might be of help:

- **adoptr** online documentation at https://kkmann.github.io/adoptr/
- paper on the theoretical background of the core **adoptr** functionality (Pilz et al., 2019)
- a general overview on adaptive designs is given in (Bauer et al., 2015)
- a more extensive treatment of the subject in (Wassmer and Brannath, 2016).

1.2 Scope

adoptr itself already makes extensive use of unittesting to ensure correctness of all implemented functions. Yet, due to constraints on the build-time for an R package, the range of scenarios covered in the unittests of **adoptr** is rather limited. Furthermore, the current R unittesting framework does not permit an easy generation of a human-readable report of the test cases to acertain coverage and test quality. Therefore, **adoptr** splits testing in two parts: technical correctness is ensured via an extensive unittesting suit in **adoptr** itself (aiming to maintain a 100% code coverage).

The validation report, however, runs through a wide range of possible application scenarios and ensures plausibility of results as well as consistency with existing methods wherever possible. The report itself is implemented as a collection of Rmarkdown documents allowing to show both the underlying code as well as the corresponding output in a human-readable format.

The online version of the report is dynamically re-generated on a daily basis using the Travis-CI service based on the respective most current version of **adoptr** on CRAN. The result of this daily build is available at https://kkmann.github.io/adoptr-validation-report/. To ensure early warning in case of any test-case failures, formal tests are implemented using the **testthat** package (Wickham et al., 2018). I.e., the combination of using a unittesting framework, a continuous integration, and continuous deployment service leads to an always up-to-date validation report (build on the current R release on Linux). Any failure of the integrated formal tests will cause the build status of the validation report to switch from 'passing' to 'failed' and the respective maintainer will be notified immediately.

1.2.1 Validating a local installation of adoptr

Note that, strictly speaking, the online version of the validation report only provides evidence of the correctness on the respective Travis-CI cloud virtual machine infrastructure using the respective most recent release of R and the most recent versions of the dependencies available on CRAN. In some instances it might therefore be desireable to conduct a local validaton of **adoptr**.

To do so, one should install **adoptr** with the `INSTALL_opts` option to include tests and invoke the test suit locally via

```r
install.packages("adoptr", INSTALL_opts = c("--install-tests"))
tools::testInstalledPackage("adoptr", types = c("examples", "tests"))
```

Upon passing the test suit successfully, the validation report can be build locally. To do so, first clone the entire source directory and switch to the newly created folder

```
git clone https://github.com/kkmann/adoptr-validation-report.git
cd adoptr-validation-report
```

Make sure that all packages requied for building the report are available, i.e., install all dependencies listed in the top-level `DESCRIPTION` file, e.g.,

```r
install.packages(c(
    "adoptr",
    "tidyverse",
    "bookdown",
    "rpact",
    "testthat",
    "pwr" ) )
```

The book can then be build using the terminal command

```
Rscript -e 'bookdown::render_book("index.Rmd", output_format = "all")'
```

or directly from R via

```r
bookdown::render_book("index.Rmd", output_format = "all")
```

This produces a new folder `_book` with the html and pdf versions of the report.

1.3 Validation Scenarios

1.3.1 Scenario I: Large effect, point prior

This is the default scenario.

- **Data distribution:** Two-armed trial with normally distributed test statistic
- **Prior:** $\delta \sim \delta_{0.4}$
- **Null hypothesis:** $\mathcal{H}_0 : \delta \leq 0$

1.3.1.1 Variant I.1: Minimizing Expected Sample Size under the Alternative

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\,|\,\delta = 0.4\big]$
- **Constraints:**
    1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.4\big] \geq 0.8$
    2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.0\big] \leq 0.025$
    3. Three variants: two-stage, group-sequential, one-stage.
- **Formal tests:**
    1. Number of iterations are checked against default maximum to ensure proper convergence.
    2. All three **adoptr** variants (two-stage, group-sequential, one-stage) comply with constraints. Internally validated by testing vs. simulated values of the power curve at respective points.
    3. Is $n()$ of the optimal two-stage design monotonously decreasing on continuation area?

4. ESS of optimal two-stage design is lower than ESS of optimal group-sequential one and that is in turn lower than the one of the optimal one-stage design.
5. ESS of optimal group-sequential design is lower than ESS of externally computed group-sequential design using the rpact package.
6. Are the ESS values obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?

1.3.1.2 Variant I.2: Minimizing Expected Sample Size under the Null Hypothesis

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\,|\,\delta = 0.0\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.4\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.0\big] \leq 0.025$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Validate constraint compliance by testing vs. simulated values of the power curve at respective points.
  3. $n()$ of optimal design is monotonously increasing on continuation area.
  4. ESS of optimal two-stage design is lower than ESS of externally computed group-sequential design using the rpact package.
  5. Are the ESS values obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?

1.3.1.3 Variant I.3: Condtional Power Constraint

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\,|\,\delta = 0.4\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.4\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.0\big] \leq 0.025$
  3. $CP := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.4, X_1 = x_1\big] \geq 0.7$ for all $x_1 \in (c_1^f, c_1^e)$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Check $Power$ and $TOER$ constraints with simulation. Check CP constraint on three different values of x_1 in (c_1^f, c_1^e)
  3. Are the CP values at the three test-pivots obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?
  4. Is ESS of optimal two-stage design with CP constraint higher than ESS of optimal two-stage design without this constraint?

1.3.2 Scenario II: Large effect, Gaussian prior

Similar in scope to Scenario I, but with a continuous Gaussian prior on δ.

- **Data distribution:** Two-armed trial with normally distributed test statistic
- **Prior:** $\delta \sim \mathcal{N}(0.4, .3)$
- **Null hypothesis:** $\mathcal{H}_0 : \delta \leq 0$

1.3.2.1 Variant II.1: Minimizing Expected Sample Size

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta > 0.0\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
  3. Three variants: two-stage, group-sequential, one-stage.
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. All designs comply with type one error rate constraints (tested via simulation).
  3. ESS of optimal two-stage design is lower than ESS of optimal group-sequential one and that is in turn lower than the one of the optimal one-stage design.

1.3.2.2 Variant II.2: Minimizing Expected Sample Size under the Null hypothesis

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1) \,|\, \delta \leq 0\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta > 0.0\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Does the design comply with $TOER$ constraint (via simulation)?
  3. Is ESS lower than expected sample size under the null hypothesis for the optimal two stage design from Variant II-1?

1.3.2.3 Variant II.3: Condtional Power Constraint

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta > 0.0\big] \geq 0.8$

2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
3. $CP := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta > 0.0, X_1 = x_1\big] \geq 0.7$ for all $x_1 \in (c_1^f, c_1^e)$

- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Check $TOER$ constraint with simulation.
  3. Check CP constraint on three different values of x_1 in (c_1^f, c_1^e)
  4. Is ESS of optimal two-stage design with CP constraint higher than ESS of optimal two-stage design without the constraint?

1.3.3 Scenario III: Large effect, uniform prior

- **Data distribution:** Two-armed trial with normally distributed test statistic
- **Prior:** sequence of uniform distributions $\delta \sim \text{Unif}(0.4 - \Delta_i, 0.4 + \Delta_i)$ around 0.4 with $\Delta_i = (3 - i)/10$ for $i = 0 \ldots 3$. I.e., for $\Delta_3 = 0$ reduces to a point prior on $\delta = 0.4$.
- **Null hypothesis:** $\mathcal{H}_0 : \delta \leq 0$

1.3.3.1 Variant III.1: Convergence under Prior Concentration

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta > 0.0\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Simulated type one error rate is compared to $TOER$ constraint for each design.
  3. ESS decreases with prior variance.

Additionally, the designs are compared graphically. Inspect the plot to see convergence pattern.

1.3.4 Scenario IV: Smaller effect size, larger trials

1.3.4.1 Variant IV.1: Minimizing Expected Sample Size under the Alternative

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1) \,|\, \delta = 0.2\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.2\big] \geq 0.8$

2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
3. Three variants: two-stage, group-sequential, one-stage.

- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. All three adoptr variants (two-stage, group-sequential, one-stage) comply with constraints. Internally validated by testing vs. simulated values of the power curve at respective points.
  3. ESS of optimal two-stage design is lower than ESS of optimal group-sequential one and that is in turn lower than the one of the optimal one-stage design.
  4. ESS of optimal group-sequential design is lower than ESS of externally computed group-sequential design using the rpact package.
  5. Are the ESS values obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?
  6. Is $n()$ of the optimal two-stage design monotonously decreasing on continuation area?

1.3.4.2 Variant IV.2: Increasing Power

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1) \,|\, \delta = 0.2\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.2\big] \geq 0.9$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
  3. Three variants: two-stage, group-sequential, one-stage.
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Does the design respect all constraints (via simulation)?
  3. ESS of optimal two-stage design is lower than ESS of optimal group-sequential one and that is in turn lower than the one of the optimal one-stage design.
  4. ESS of optimal group-sequential design is lower than ESS of externally computed group-sequential design using the rpact package.
  5. Are the ESS values obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?
  6. Is $n()$ of the optimal two-stage design monotonously decreasing on continuation area?

1.3.4.3 Variant IV.3: Increasing Maximal Type One Error Rate

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1) \,|\, \delta = 0.2\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.2\big] \geq 0.8$

2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.05$
3. Three variants: two-stage, group-sequential, one-stage.

- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Does the design respect all constraints (via simulation)?
  3. ESS of optimal two-stage design is lower than ESS of optimal group-sequential one and that is in turn lower than the one of the optimal one-stage design.
  4. ESS of optimal group-sequential design is lower than ESS of externally computed group-sequential design using the rpact package.
  5. Are the ESS values obtained from simulation the same as the ones obtained by using numerical integration via `adoptr::evaluate`?
  6. Is $n()$ of the optimal two-stage design monotonously decreasing on continuation area?

1.3.5 Scenario V: Single-arm design, medium effect size

- **Data distribution:** One-armed trial with normally distributed test statistic
- **Prior:** $\delta \sim \delta_{0.3}$
- **Null hypothesis:** $\mathcal{H}_0 : \delta \leq 0$

1.3.5.1 Variant V.1: Sensitivity to Integration Order

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1) \,|\, \delta = 0.3\big]$
- **Constraints:**
  1. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.3\big] \geq 0.8$
  2. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
  3. Three variants: integration order 5, 8, 11 two-stage designs.
- **Formal tests:**
  1. Do all designs converge within the respective iteration limit?
  2. Do all designs respect all constraints (via simulation)?

1.3.5.2 Variant V.2: Utility Maximization

- **Objective:** $\lambda\, Power - ESS := \lambda\, \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.3\big] - \boldsymbol{E}\big[n(X_1) \,|\, \delta = 0.3\big]$. for $\lambda = 200$ and 500
- **Constraints:**
  1. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2 \,|\, \delta = 0.0\big] \leq 0.025$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.

2. Do both designs respect the type one error rate constraint (via simulation)?
3. Is the power of the design with larger λ larger?

1.3.5.3 Variant V.3: n_1 penalty

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\,|\,\delta = 0.3\big] + \lambda\, n_1$ for $\lambda = 0.05$ and 0.2.
- **Constraints:**
  1. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.0\big] \leq 0.025$
  2. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.3\big] \geq 0.8$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Is n_1 for the optimal design smaller than the order-5 design in V.1?

1.3.5.4 Variant V.4: n_2 penalty

- **Objective:** $ESS := \boldsymbol{E}\big[n(X_1)\,|\,\delta = 0.3\big] + \lambda$ `AverageN2` for $\lambda = 0.01$ and 0.1.
- **Constraints:**
  1. $TOER := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.0\big] \leq 0.025$
  2. $Power := \boldsymbol{Pr}\big[c_2(X_1) < X_2\,|\,\delta = 0.3\big] \geq 0.8$
- **Formal tests:**
  1. Number of iterations are checked against default maximum to ensure proper convergence.
  2. Is the `AverageN2` for the optimal design smaller than for the order-5 design in V.1?

1.4 Technical Setup

All scenarios are run in a single, shared R session. Required packages are loaded here, the random seed is defined and set centrally, and the default number of iteration is increased to make sure that all scenarios converge properly. Additionally R scripts with convenience functions are sourced here as well. There are three additional functions for this report. `rpact_design` creates a two-stage design via the package **rpact** (Wassmer and Pahlke, 2018) in the notation of **adoptr**. `sim_pr_reject` and `sim_n` allow to simulate rejection probabilities and expected sample sizes respectively by the **adoptr** routine `simulate`. Furthermore, global tolerances for the validation are set. For error rates, a relative deviation of 1% from the target value is accepted. (Expected) Sample sizes deviations are more liberally accepted up to 0.5.

```r
library(adoptr)
library(tidyverse)
library(rpact)
library(pwr)
library(testthat)

# load custom functions in folder subfolder '/R'
for (nm in list.files("R", pattern = "\\.[RrSsQq]$"))
   source(file.path("R", nm))

# define seed value
seed  <- 42

# define absolute tolerance for error rates
tol   <- 0.01

# define absolute tolerance for sample sizes
tol_n <- 0.5

# define custom tolerance and iteration limit for nloptr
opts = list(
    algorithm = "NLOPT_LN_COBYLA",
    xtol_rel  = 1e-5,
    maxeval   = 100000
)
```

Chapter 2

Scenario I: large effect, point prior

2.1 Details

In this scenario, a classical two-arm trial with normal test statistic and known variance (w.l.o.g. variance of the test statistic is 1). This situation corresponds to a classical z-test for a difference in population means. The null hypothesis is no population mean difference, i.e. $\mathcal{H}_0 : \delta \leq 0$. An alternative effect size of $\delta = 0.4$ with point prior distribution is assumed. Across all variants in this scenario, the one-sided maximal type one error rate is restricted to $\alpha = 0.025$ and the power at the point alternative of $\delta = 0.4$ must be at least 0.8.

```
# data distribution and hypotheses
datadist   <- Normal(two_armed = TRUE)
H_0        <- PointMassPrior(.0, 1)
prior      <- PointMassPrior(.4, 1)

# define constraints
alpha      <- 0.025
min_power  <- 0.8
toer_cnstr <- Power(datadist, H_0)   <= alpha
pow_cnstr  <- Power(datadist, prior) >= min_power
```

2.2 Variant I-1: Minimizing Expected Sample Size under Point Prior

2.2.1 Objective

Firstly, expected sample size under the alternative (point prior) is minimized, i.e., $\boldsymbol{E}\big[n(\mathcal{D})\big]$.

```
ess <- ExpectedSampleSize(datadist, prior)
```

2.2.2 Constraints

No additional constraints besides type one error rate and power are considered in this variant.

2.2.3 Initial Designs

For this example, the optimal one-stage, group-sequential, and generic two-stage designs are computed. The initial design for the one-stage case is determined heuristically (cf. Scenario III where another initial design is applied on the same situation for stability of initial values). Both the group sequential and the generic two-stage designs are optimized starting from the corresponding group-sequential design as computed by the `rpact` package.

```
order <- 7L
# data frame of initial designs
tbl_designs <- tibble(
    type    = c("one-stage", "group-sequential", "two-stage"),
    initial = list(
        OneStageDesign(200, 2.0),
        rpact_design(0.4, 0.025, 0.8, TRUE, order),
        TwoStageDesign(rpact_design(0.4, 0.025, 0.8, TRUE, order))) )
```

The order of integration is set to 7.

2.2.4 Optimization

```
tbl_designs <- tbl_designs %>%
    mutate(
       optimal = purrr::map(initial, ~minimize(

          ess,
```

```
          subject_to(
              toer_cnstr,
              pow_cnstr
          ),

          initial_design = .,
          opts           = opts)) )
```

2.2.5 Test Cases

To avoid improper solutions, it is first verified that the maximum number of iterations was not exceeded in any of the three cases.

```
tbl_designs %>%
  transmute(
      type,
      iterations = purrr::map_int(tbl_designs$optimal,
                                  ~.$nloptr_return$iterations) ) %>%
  {print(.); .} %>%
  {testthat::expect_true(all(.$iterations < opts$maxeval))}
```

```
## # A tibble: 3 x 2
##   type             iterations
##   <chr>                 <int>
## 1 one-stage                24
## 2 group-sequential       1380
## 3 two-stage              4140
```

Next, the type one error rate and power constraints are verified for all three designs by simulation:

```
tbl_designs %>%
  transmute(
      type,
      toer  = purrr::map(tbl_designs$optimal,
                         ~sim_pr_reject(.[[1]], .0, datadist)$prob),
      power = purrr::map(tbl_designs$optimal,
                         ~sim_pr_reject(.[[1]], .4, datadist)$prob) ) %>%
  unnest() %>%
  {print(.); .} %>% {
  testthat::expect_true(all(.$toer  <= alpha * (1 + tol)))
  testthat::expect_true(all(.$power >= min_power * (1 - tol))) }
```

```
## # A tibble: 3 x 3
##   type               toer power
##   <chr>             <dbl> <dbl>
```

```
## 1 one-stage        0.0251 0.799
## 2 group-sequential 0.0250 0.800
## 3 two-stage        0.0250 0.799
```

The n_2 function of the optimal two-stage design is expected to be monotonously decreasing:

```
expect_true(
    all(diff(
        # get optimal two-stage design n2 pivots
        tbl_designs %>% filter(type == "two-stage") %>%
           {.[["optimal"]][[1]]$design@n2_pivots}
        ) < 0) )
```

Since the degrees of freedom of the three design classes are ordered as 'two-stage' > 'group-sequential' > 'one-stage', the expected sample sizes (under the alternative) should be ordered in reverse ('two-stage' smallest). Additionally, expected sample sizes under both null and alternative are computed both via `evaluate()` and simulation-based.

```
ess0 <- ExpectedSampleSize(datadist, H_0)

tbl_designs %>%
    mutate(
        ess      = map_dbl(optimal,
                       ~evaluate(ess, .$design) ),
        ess_sim  = map_dbl(optimal,
                       ~sim_n(.$design, .4, datadist)$n ),
        ess0     = map_dbl(optimal,
                       ~evaluate(ess0, .$design) ),
        ess0_sim = map_dbl(optimal,
                       ~sim_n(.$design, .0, datadist)$n ) ) %>%
    {print(.); .} %>% {
    # sim/evaluate same under alternative?
    testthat::expect_equal(.$ess, .$ess_sim,
                           tolerance = tol_n,
                           scale = 1)
    # sim/evaluate same under null?
    testthat::expect_equal(.$ess0, .$ess0_sim,
                           tolerance = tol_n,
                           scale = 1)
    # monotonicity with respect to degrees of freedom
    testthat::expect_true(all(diff(.$ess) < 0)) }
```

```
## # A tibble: 3 x 7
##   type             initial    optimal      ess ess_sim  ess0 ess0_sim
##   <chr>            <list>     <list>     <dbl>   <dbl> <dbl>    <dbl>
## 1 one-stage        <OnStgDsg> <list [3]>  98      98    98       98
```

```
## 2 group-sequential <GrpSqntD> <list [3]>  80.9    80.9  68.5     68.5
## 3 two-stage        <TwStgDsg> <list [3]>  79.6    79.7  68.9     68.9
```

2.3 Variant I-2: Minimizing Expected Sample Size under Null Hypothesis

2.3.1 Objective

Expected sample size under the null hypothesis prior is minimized, i.e., `ess0`.

2.3.2 Constraints

The constraints remain unchanged from the base case.

2.3.3 Initial Design

Since optimization under the null favours an entirely different (monotonically increasing) sample size function, and thus also a different shape of the c_2 function, the `rpact` initial design is a suboptimal starting point. Instead, we start with a constant c_2 function by heuristically setting it to 2 on the continuation area. Also, optimizing under the null favours extremely conservative boundaries for early efficacy stopping and we thus impose as fairly liberal upper bound of 3 for early efficacy stopping.

```
init_design_h0 <- tbl_designs %>%
    filter(type == "two-stage") %>%
    pull(initial) %>%
    .[[1]]
init_design_h0@c2_pivots <- rep(2, order)

ub_design <- TwoStageDesign(
    3*init_design_h0@n1,
    2,
    3,
    rep(300, order),
    rep(3.0, order)
)
```

2.3.4 Optimization

The optimal two-stage design is computed.

```
opt_h0 <- minimize(

    ess0,

    subject_to(
        toer_cnstr,
        pow_cnstr
    ),

    initial_design        = init_design_h0,
    upper_boundary_design = ub_design,
    opts = opts )
```

2.3.5 Test Cases

Make sure that the optimization algorithm converged within the set maximum number of iterations:

```
opt_h0$nloptr_return$iterations %>%
    {print(.); .} %>%
    {testthat::expect_true(. < opts$maxeval)}
```

```
## [1] 17876
```

The n_2 function of the optimal two-stage design is expected to be monotnously increasing.

```
expect_true(
    all(diff(opt_h0$design@n2_pivots) > 0) )
```

Next, the type one error rate and power constraints are tested.

```
tbl_performance <- tibble(
    delta = c(.0, .4) ) %>%
    mutate(
        power     = map(
            delta,
            ~evaluate(
                Power(datadist, PointMassPrior(., 1)),
                opt_h0$design) ),
        power_sim = map(
            delta,
            ~sim_pr_reject(opt_h0$design, ., datadist)$prob),
        ess       = map(
            delta,
            ~evaluate(ExpectedSampleSize(
```

```
                    datadist,
                    PointMassPrior(., 1) ),
                opt_h0$design) ),
        ess_sim   = map(
            delta,
            ~sim_n(opt_h0$design, . ,datadist)$n ) ) %>%
    unnest

print(tbl_performance)
```

```
## # A tibble: 2 x 5
##   delta  power power_sim   ess ess_sim
##   <dbl>  <dbl>     <dbl> <dbl>   <dbl>
## 1   0   0.0250    0.0250  57.2    57.3
## 2   0.4 0.800     0.802  117.    118.
```

```
testthat::expect_lte(
    tbl_performance %>% filter(delta == 0) %>% pull(power_sim),
    alpha * (1 + tol) )

testthat::expect_gte(
    tbl_performance %>% filter(delta == 0.4) %>% pull(power_sim),
    min_power * (1 - tol) )

# make sure that evaluate() leads to same results
testthat::expect_equal(
    tbl_performance$power, tbl_performance$power_sim,
    tol   = tol,
    scale = 1 )

testthat::expect_equal(
    tbl_performance$ess, tbl_performance$ess_sim,
    tol   = tol_n,
    scale = 1 )
```

The expected sample size under the null must be lower or equal than the expected sample size of the inital `rpact` group-sequential design.

```
testthat::expect_gte(
    evaluate(ess0,
             tbl_designs %>%
                filter(type == "two-stage") %>%
                pull(initial) %>%
                .[[1]] ),
    evaluate(ess0, opt_h0$design) )
```

2.4 Variant I-3: Conditional Power Constraint

2.4.1 Objective

Same as in I-1, i.e., expected sample size under the alternative point prior is minimized.

2.4.2 Constraints

Besides the previous global type one error rate and power constraints, an additional constraint on *conditional* power is imposed.

```
cp       <- ConditionalPower(datadist, prior)
cp_cnstr <- cp >= .7
```

2.4.3 Initial Design

The same initial (generic two-stage) design as in I-1 is used.

2.4.4 Optimization

```
opt_cp <- minimize(

    ess,
    subject_to(
        toer_cnstr,
        pow_cnstr,
        cp_cnstr # new constraint
    ),

    initial_design = tbl_designs %>%
        filter(type == "two-stage") %>%
        pull(initial) %>%
        .[[1]],
    opts = opts )
```

2.4.5 Test Cases

Check if the optimization algorithm converged.

```
opt_cp$nloptr_return$iterations %>%
    {print(.); .} %>%
    {testthat::expect_true(. < opts$maxeval)}
```

```
## [1] 4059
```

Check constraints.

```
tbl_performance <- tibble(
    delta = c(.0, .4) ) %>%
    mutate(
        power     = map(
            delta,
            ~evaluate(
                Power(datadist, PointMassPrior(., 1)),
                opt_cp$design) ),
        power_sim = map(
            delta,
            ~sim_pr_reject(opt_cp$design, ., datadist)$prob),
        ess       = map(
            delta,
            ~evaluate(ExpectedSampleSize(
                    datadist,
                    PointMassPrior(., 1) ),
                opt_cp$design) ),
        ess_sim   = map(
            delta,
            ~sim_n(opt_cp$design, . ,datadist)$n ) ) %>%
    unnest

print(tbl_performance)
```

```
## # A tibble: 2 x 5
##   delta  power power_sim   ess ess_sim
##   <dbl>  <dbl>     <dbl> <dbl>   <dbl>
## 1   0    0.0250    0.0250  68.9    69.0
## 2   0.4 0.800     0.799   79.7    79.8
```

```
testthat::expect_lte(
    tbl_performance %>% filter(delta == 0) %>% pull(power_sim),
    alpha * (1 + tol) )

testthat::expect_gte(
    tbl_performance %>% filter(delta == 0.4) %>% pull(power_sim),
    min_power * (1 - tol) )

# make sure that evaluate() leads to same results
```

```
testthat::expect_equal(
    tbl_performance$power, tbl_performance$power_sim,
    tol   = tol,
    scale = 1 )

testthat::expect_equal(
    tbl_performance$ess, tbl_performance$ess_sim,
    tol   = tol_n,
    scale = 1 )
```

The conditional power constraint is evaluated and tested on a grid over the continuation region (both simulated an via numerical integration).

```
tibble(
    x1     = seq(opt_cp$design@c1f, opt_cp$design@c1e, length.out = 25),
    cp     = map_dbl(x1, ~evaluate(cp, opt_cp$design, .)),
    cp_sim = map_dbl(x1, function(x1) {
        x2  <- simulate(datadist, 10^6, n2(opt_cp$design, x1), .4, 42)
        rej <- ifelse(x2 > c2(opt_cp$design, x1), 1, 0)
        return(mean(rej))
    }) ) %>%
  {print(.); .} %>% {
      testthat::expect_true(all(.$cp     >= 0.7 * (1 - tol)))
      testthat::expect_true(all(.$cp_sim >= 0.7 * (1 - tol)))
      testthat::expect_true(all(abs(.$cp - .$cp_sim) <= tol)) }
```

```
## # A tibble: 25 x 3
##       x1    cp cp_sim
##    <dbl> <dbl>  <dbl>
##  1 0.810 0.701  0.701
##  2 0.872 0.698  0.699
##  3 0.934 0.701  0.701
##  4 0.995 0.698  0.698
##  5 1.06  0.700  0.700
##  6 1.12  0.702  0.703
##  7 1.18  0.699  0.699
##  8 1.24  0.703  0.703
##  9 1.30  0.713  0.713
## 10 1.37  0.718  0.718
## # ... with 15 more rows
```

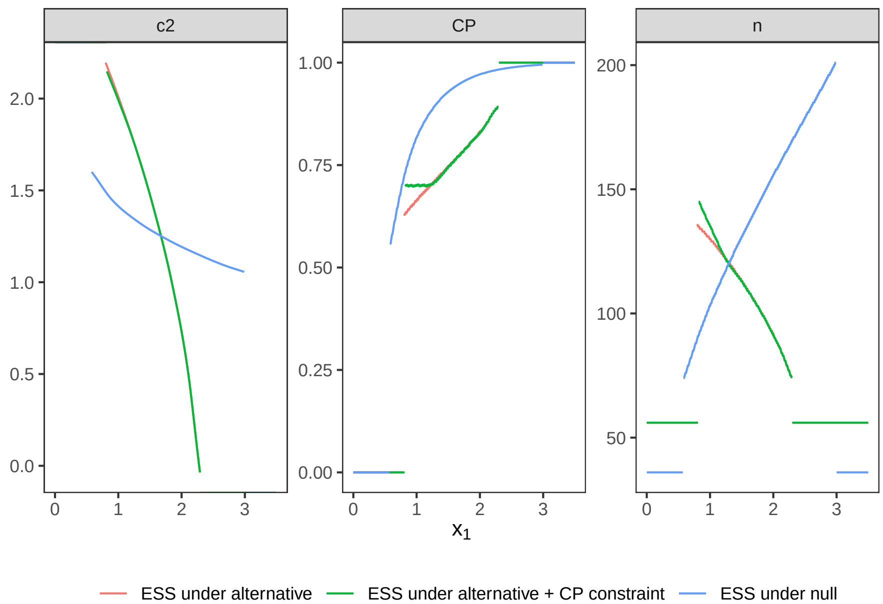
Finally, the expected sample size under the alternative prior should be larger than in the case without the constraint I-1.

```
testthat::expect_gte(
    evaluate(ess, opt_cp$design),
    evaluate(
```

```
        ess,
        tbl_designs %>%
            filter(type == "two-stage") %>%
            pull(optimal) %>%
            .[[1]] %>%
            .$design ) )
```

2.5 Plot Two-Stage Designs

The following figure shows the three optimal two-stage designs side by side. The effect of the conditional power constraint (CP not below 0.7) is clearly visible and the very different characteristics between optimizing under the null or the alternative are clearly visible.



Chapter 3

Scenario II: Large effect, Gaussian prior

3.1 Details

In this scenario, we revisit the case from Scenario I, but are not assuming a point prior any more. Instead, a Gaussian prior with mean $\vartheta = 0.4$ and variance $\tau^2 = 0.2^2$ on the effect size is assumed, i.e. $\delta \sim \mathcal{N}(0.4, 0.2^2)$.

In order to fulfill regulartory considerations, the type one error rate is still protected under the point prior $\delta = 0$ at the level of significance $\alpha = 0.025$.

The power constraint, however, needs to be modified. It is not senseful to compute the power as rejection probability under the full prior, because effect sizes less than a minimal clinically relevant effect do not show (sufficient) evidence againt the null hypothesis. Therefore, we assume a minimal clinically relevant effect size $\delta = 0.0$ and condition the prior to compute expected power. In the following, the expected power should be at least 0.8.

```
# data distribution and priors
datadist   <- Normal(two_armed = TRUE)
H_0        <- PointMassPrior(.0, 1)
prior      <- ContinuousPrior(function(delta) dnorm(delta, mean = .4, sd = .2),
                              support = c(-5, 5),
                              tighten_support = TRUE)

# define constraints on type one error rate and expected power
alpha      <- 0.025
min_epower <- 0.8
toer_cnstr <- Power(datadist, H_0) <= alpha
```

```
epow_cnstr <- Power(datadist, condition(prior, c(0.0, prior@support[2]))) >= min_epowe
```

3.2 Variant II-1: Minimizing Expected Sample Size under Point Prior

3.2.1 Objective

Expected sample size under the full prior is minimized, i.e., $\boldsymbol{E}\big[n(\mathcal{D})\big]$.

```
ess <- ExpectedSampleSize(datadist, prior)
```

3.2.2 Constraints

No additional constraints are considered in this variant.

3.2.3 Initial Design

For this example, the optimal one-stage, group-sequential, and generic two-stage designs are computed. While the initial design for the one-stage case is determined heuristically, both the group sequential and the generic two-stage designs are optimized starting from the a group-sequential design that is computed by the **rpact** package to fulfill the type one error rate constraint and that fulfills the power constraint at an effect size of $\delta = 0.3$.

```
order <- 5L
# data frame of initial designs
tbl_designs <- tibble(
    type    = c("one-stage", "group-sequential", "two-stage"),
    initial = list(
        OneStageDesign(250, 2.0),
        rpact_design(0.3, 0.025, 0.8, TRUE, order),
        TwoStageDesign(rpact_design(0.3, 0.025, 0.8, TRUE, order))) )
```

The order of integration is set to 5.

3.2.4 Optimization

For all these three initial designs, the resulting optimal designs are computed.

```
tbl_designs <- tbl_designs %>%
    mutate(
```

```
    optimal = purrr::map(initial, ~minimize(

        ess,
        subject_to(
            toer_cnstr,
            epow_cnstr
        ),

        initial_design = .,
        opts           = opts)) )
```

3.2.5 Test Cases

Firstly, it is checked that the maximum number of iterations was not reached in all these cases.

```
tbl_designs %>%
  transmute(
      type,
      iterations = purrr::map_int(tbl_designs$optimal,
                                  ~.$nloptr_return$iterations) ) %>%
  {print(.); .} %>%
  {testthat::expect_true(all(.$iterations < opts$maxeval))}
```

```
## # A tibble: 3 x 2
##   type            iterations
##   <chr>                <int>
## 1 one-stage               21
## 2 group-sequential       670
## 3 two-stage             1928
```

Since type one error rate is defined under the point effect size $\delta = 0$, the type one error rate constraint can be tested for all three optimal designs.

```
tbl_designs %>%
  transmute(
      type,
      toer = purrr::map(tbl_designs$optimal,
                        ~sim_pr_reject(.[[1]], .0, datadist)$prob) ) %>%
  unnest() %>%
  {print(.); .} %>% {
  testthat::expect_true(all(.$toer <= alpha * (1 + tol))) }
```

```
## # A tibble: 3 x 2
##   type              toer
##   <chr>            <dbl>
```

```
## 1 one-stage        0.0251
## 2 group-sequential 0.0249
## 3 two-stage        0.0250
```

Since the optimal two-stage design is more flexible than the optimal group-sequential design (constant n_2 function) and this is more flexible than the optimal one-stage design (no second stage), the expected sample sizes under the prior should be ordered in the opposite way. Additionally, expected sample sizes under the null hypothesis is computed both via `evaluate()` and simulation-based.

```
essh0 <- ExpectedSampleSize(datadist, H_0)

tbl_designs %>%
    mutate(
        ess       = map_dbl(optimal,
                            ~evaluate(ess, .$design) ),
        essh0     = map_dbl(optimal,
                            ~evaluate(essh0, .$design) ),
        essh0_sim = map_dbl(optimal,
                            ~sim_n(.$design, .0, datadist)$n ) ) %>%
    {print(.); .} %>% {
    # sim/evaluate same under null?
    testthat::expect_equal(.$essh0, .$essh0_sim,
                           tolerance = tol_n,
                           scale = 1)
    # monotonicity with respect to degrees of freedom
    testthat::expect_true(all(diff(.$ess) < 0)) }
```

```
## # A tibble: 3 x 6
##   type             initial    optimal      ess essh0 essh0_sim
##   <chr>            <list>     <list>     <dbl> <dbl>     <dbl>
## 1 one-stage        <OnStgDsg> <list [3]>  165   165       165
## 2 group-sequential <GrpSqntD> <list [3]>  115.  110.      110.
## 3 two-stage        <TwStgDsg> <list [3]>  113.  114.      114.
```

3.3 Variant II-2: Minimizing Expected Sample Size under Null Hypothesis

3.3.1 Objective

Expected sample size conditioned on negative effect sizes is minimized, i.e.,

```
ess_0 <- ExpectedSampleSize(datadist, condition(prior, c(bounds(prior)[1], 0)))
```

3.3.2 Constraints

No additional constraints besides type one error rate and expected power are considered in this variant.

3.3.3 Initial Design

As in Variant I.2 another initial design is more appropriate for optimization under the null hypothesis. In this situation, one may expect a different (increasing) sample size function, and thus also a different shape of the c_2 function. Therefore, the `rpact` initial design is a suboptimal starting point. Instead, we start with a constant c_2 function by heuristically setting it to 2 on the continuation area.

```
init_design_h0 <- tbl_designs %>%
    filter(type == "two-stage") %>%
    pull(initial) %>%
    .[[1]]
init_design_h0@c2_pivots <- rep(2, order)
```

3.3.4 Optimization

The optimal two-stage design is computed.

```
opt_neg <- minimize(

        ess_0,

        subject_to(

            toer_cnstr,
            epow_cnstr
        ),

        initial_design = init_design_h0,
        opts = opts
)
```

3.3.5 Test Cases

First of all, check if the optimization algorithm converged. To avoid improper solutions, it is first verified that the maximum number of iterations was not exceeded in any of the three cases.

```
testthat::expect_true(opt_neg$nloptr_return$iterations < opts$maxeval)
print(opt_neg$nloptr_return$iterations)
```

```
## [1] 1025
```

Again, the type one error rate under the point null hypothesis $\delta = 0$ can be tested by simulation.

```
tbl_toer <- tibble(
  toer     = evaluate(Power(datadist, H_0), opt_neg$design),
  toer_sim = sim_pr_reject(opt_neg$design, .0, datadist)$prob
)

print(tbl_toer)
```

```
## # A tibble: 1 x 2
##     toer toer_sim
##    <dbl>    <dbl>
## 1 0.0250   0.0250
```

```
testthat::expect_true(tbl_toer$toer <= alpha * (1 + tol))
testthat::expect_true(tbl_toer$toer_sim <= alpha * (1 + tol))
```

Furthermore, the expected sample size under the prior conditioned on negative effect sizes ($\delta \leq 0$) should be lower for the optimal design derived in this variant than for the optimal design from Variant II.1 where expected sample size under the full prior was minimized.

```
testthat::expect_lte(
    evaluate(ess_0, opt_neg$design),
    evaluate(
        ess_0,
        tbl_designs %>%
            filter(type == "two-stage") %>%
            pull(optimal) %>%
            .[[1]] %>%
            .$design )
)
```

3.4 Variant II-3: Conditional Power Constraint

3.4.1 Objective

As in Variant II.1, expected sample size under the full prior is minimized.

3.4.2 Constraints

In addition to the constraints on type one error rate and expected power, a constraint on conditional power to be larger than 0.7 is included.

```
cp <- ConditionalPower(datadist, condition(prior, c(0, prior@support[2])))

cp_cnstr <- cp >= 0.7
```

3.4.3 Initial Design

The previous initial design can still be applied.

3.4.4 Optimization

The optimal two-stage design is computed.

```
opt_cp <- minimize(
        ess,
        subject_to(
            toer_cnstr,
            epow_cnstr,
            cp_cnstr
        ),
        initial_design = tbl_designs$initial[[3]],
        opts = opts
)
```

3.4.5 Test Cases

We start checking whether the maximum number of iterations was not reached.

```
print(opt_cp$nloptr_return$iterations)
```

```
## [1] 2154
```

```
testthat::expect_true(opt_cp$nloptr_return$iterations < opts$maxeval)
```

The type one error rate is tested via simulation and compared to the value obtained by `evaluate()`.

```
tbl_toer <- tibble(
  toer     = evaluate(Power(datadist, H_0), opt_cp$design),
  toer_sim = sim_pr_reject(opt_cp$design, .0, datadist)$prob
)
```

```
print(tbl_toer)
```

```
## # A tibble: 1 x 2
##     toer toer_sim
##    <dbl>    <dbl>
## 1 0.0250   0.0250
```

```
testthat::expect_true(tbl_toer$toer <= alpha * (1 + tol))
testthat::expect_true(tbl_toer$toer_sim <= alpha * (1 + tol))
```

The conditional power is evaluated via numerical integration on several points inside the continuation region and it is tested whether the constraint is fulfilled on all these points.

```
tibble(
    x1 = seq(opt_cp$design@c1f, opt_cp$design@c1e, length.out = 25),
    cp = map_dbl(x1, ~evaluate(cp, opt_cp$design, .)) ) %>%
  {print(.); .} %>% {
      testthat::expect_true(all(.$cp >= 0.7 * (1 - tol))) }
```

```
## # A tibble: 25 x 2
##       x1    cp
##    <dbl> <dbl>
##  1 0.718 0.696
##  2 0.787 0.700
##  3 0.855 0.702
##  4 0.924 0.700
##  5 0.993 0.699
##  6 1.06  0.698
##  7 1.13  0.702
##  8 1.20  0.708
##  9 1.27  0.716
## 10 1.34  0.725
## # ... with 15 more rows
```

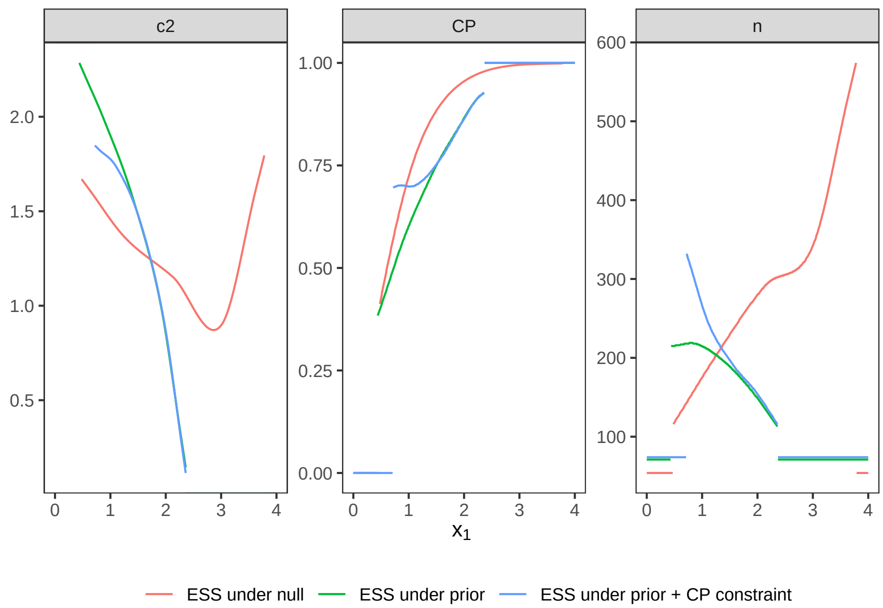
Due to the additional constraint in comparison to Variant II.1, Variant II.3 should show a larger expected sample size under the prior than Variant II.1

```
testthat::expect_gte(
    evaluate(ess, opt_cp$design),
    evaluate(
        ess,
        tbl_designs %>%
            filter(type == "two-stage") %>%
            pull(optimal) %>%
            .[[1]] %>%
            .$design )
```

```
)
```

3.5 Plot Two-Stage Designs

The optimal two-stage designs stemming from the different variants are plotted together.



Chapter 4

Scenario III: large effect, uniform prior

4.1 Details

This scenario covers a similar setting as Scenario I. The purpose is to asses whether placing uniform priors with decreasing width of support centered at the $\delta = 0.4$ leads to a sequence of optimal designs which converges towards the solution in variant I-1.

The trial is still considered to be two-armed with normally distribtuted outcomes and the type one error rate under the null hypothesis $\mathcal{H}_0 : \delta \leq 0$ is to be protected at $\alpha = 0.025$.

```
datadist   <- Normal(two_armed = TRUE)
H_0        <- PointMassPrior(.0, 1)
alpha      <- 0.025
toer_cnstr <- Power(datadist, H_0) <= alpha
```

In this scenario we consider a sequence of uniform distributions $\delta \sim \text{Unif}(0.4 - \Delta_i, 0.4 + \Delta_i)$ around 0.4 with $\Delta_i = (3 - i)/10$ for $i = 0 \ldots 3$. I.e., for $\Delta_3 = 0$ reduces to `PointMassPrior` on $\delta = 0.4$.

```
prior <- function(delta) {
    if (delta == 0)
        return(PointMassPrior(.4, 1.0))
    a <- .4 - delta; b <- .4 + delta
    ContinuousPrior(function(x) dunif(x, a, b), support = c(a, b))
}
```

Across all variants in this scenario, the expected power under the respective prior conditioned on $\delta > 0$ must be at least 0.8. I.e., throughout this scenario, we always use the following constraint on expected power.

```
ep_cnstr <- function(delta) {
    prior     <- prior(delta)
    cnd_prior <- condition(prior, c(0, bounds(prior)[2]))
    return( Power(datadist, cnd_prior) >= 0.8 )
}
```

4.2 Variant III.1: Convergence under prior concentration

The goal of this variant is to make sure that the optimal solution converges as the prior is more and more concentrated at a point mass.

4.2.1 Objective

Expected sample size under the respective prior is minimized, i.e., $\boldsymbol{E}\big[n(\mathcal{D})\big]$.

```
objective <- function(delta) {
    ExpectedSampleSize(datadist, prior(delta))
}
```

4.2.2 Constraints

The constraints have already been described under details.

4.2.3 Optimization problem

The optimization problem depending on Δ_i is defined below. The default optimization parameters, 5 pivot points, and a fixed initial design are used. The initial design is chosen such that the error constraints are fulfilled. Early stopping for futility is applied if the effect shows in the opponent direction to the alternative, i.e. $c_1^f = 0$. c_2 is chosen close to and c_1^e a little larger than the $1-\alpha$-quantile of the standard normal distribution. The sample sizes are selected to fulfill the error constraints.

```
init <- TwoStageDesign(
    n1    = 150,
    c1f   = 0,
    c1e   = 2.3,
    n2    = 125.0,
```

```
    c2    = 2.0,
    order = 5
)

optimal_design <- function(delta) {
    minimize(
        objective(delta),
        subject_to(
            toer_cnstr,
            ep_cnstr(delta)
        ),
        initial_design = init
    )
}

# compute the sequence of optimal designs
deltas  <- 3:0/10
results <- lapply(deltas, optimal_design)
```

4.2.4 Test cases

Check that iteration limit was not exceeded in any case.

```
iters <- sapply(results, function(x) x$nloptr_return$iterations)
print(iters)
```

```
## [1] 1746 1857 2438 2684
```

```
testthat::expect_true(all(iters <= opts$maxeval))
```

Check type one error rate control by simulation and by calling `evaluate()`.

```
df_toer <- tibble(
  delta    = deltas,
  toer     = sapply(results, function(x) evaluate(Power(datadist, H_0), x$design)),
  toer_sim = sapply(results, function(x) sim_pr_reject(x$design, .0, datadist)$prob)
)

testthat::expect_true(all(df_toer$toer     <= alpha * (1 + tol)))
testthat::expect_true(all(df_toer$toer_sim <= alpha * (1 + tol)))

print(df_toer)
```

```
## # A tibble: 4 x 3
##   delta   toer toer_sim
##   <dbl>  <dbl>    <dbl>
```

```
## 1   0.3 0.0250   0.0250
## 2   0.2 0.0250   0.0250
## 3   0.1 0.0250   0.0250
## 4   0   0.0250   0.0250
```

Check that expected sample size decreases with decreasing prior variance.

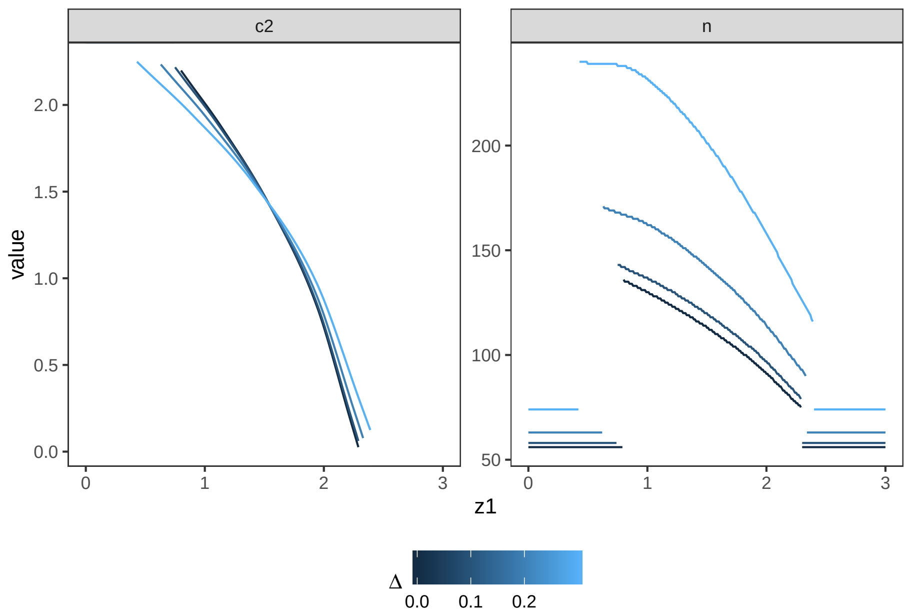
```
testthat::expect_gte(
  evaluate(objective(deltas[1]), results[[1]]$design),
  evaluate(objective(deltas[2]), results[[2]]$design)
)

testthat::expect_gte(
  evaluate(objective(deltas[2]), results[[2]]$design),
  evaluate(objective(deltas[3]), results[[3]]$design)
)

testthat::expect_gte(
  evaluate(objective(deltas[3]), results[[3]]$design),
  evaluate(objective(deltas[4]), results[[4]]$design)
)
```

4.2.5 Plot designs

Plot and assess for convergence

```
z1 <- seq(0, 3, by = .01)

tibble(
    delta  = deltas,
    design = lapply(results, function(x) x$design)
) %>%
    group_by(delta) %>%
    do(
        z1 = z1,
        n  = adoptr::n(.$design[[1]], z1),
        c2 = c2(.$design[[1]], z1)
    ) %>%
    unnest() %>%
    mutate(
        section = ifelse(
            is.finite(c2),
            "continuation",
            ifelse(c2 == -Inf, "efficacy", "futility")
        )
    ) %>%
```

```
    gather(variable, value, n, c2) %>%
    ggplot(aes(z1, value, color = delta)) +
        geom_line(aes(group = interaction(section, delta))) +
        facet_wrap(~variable, scales = "free_y") +
        theme_bw() +
        scale_color_continuous(bquote(Delta)) +
        theme(
            panel.grid = element_blank(),
            legend.position = "bottom"
        )
```



Chapter 5

Scenario IV: smaller effect, point prior

5.1 Details

In this scenario, a classical two-arm trial with normal test statistic and known variance (w.l.o.g. variance of the test statistic is 1). This situation corresponds to a classical z-test for a difference in population means.

```
datadist <- Normal(two_armed = TRUE)
```

The null hypothesis is no population mean difference, i.e. $\mathcal{H}_0 : \delta \leq 0$.

```
H_0 <- PointMassPrior(.0, 1)
```

An alternative effect size of $\delta = 0.2$ with point prior distribution is assumed.

```
prior <- PointMassPrior(.2, 1)
```

Across all variants in this scenario, the one-sided maximal type one error rate is restricted to

```
alpha <- 0.025
```

and the power at the point alternative of $\delta = 0.2$ must be at least

```
min_power <- 0.8
```

I.e. throughout this sceanrio, we always use the two constraints

```
toer_cnstr <- Power(datadist, H_0) <= alpha
```

and

```
pow_cnstr <- Power(datadist, prior) >= min_power
```

5.2 Variant IV-1: Minimizing Expected Sample Size under Point Prior

5.2.1 Objective

Expected sample size under the respective prior is minimized, i.e., $\boldsymbol{E}\big[n(\mathcal{D})\big]$.

```
ess <- ExpectedSampleSize(datadist, prior)
```

5.2.2 Constraints

No additional constraints are considered in this variant.

5.2.3 Initial Design

`adoptr` requires the definition of an initial design for optimization. We start with a group-sequential design from the package `rpact` that fulfills these constraints and is used later for comparison. The order of integration is set to 5.

```
order <- 5L

init_design_gs <- rpact_design(0.2, 0.025, 0.8, TRUE, order)
```

5.2.4 Optimization

The optimal design is computed in three variants: two-stage, group-sequential and one-stage. The input only differs with regard to the initial design. The optimal group-sequential design is used as initial design to compute the optimal two-stage design.

```
opt_design <- function(initial_design) {
    minimize(
        ess,
        subject_to(
            toer_cnstr,
            pow_cnstr
        ),
        initial_design = initial_design,
        opts = opts
```

```
    )
}

opt1_gs <- opt_design(init_design_gs)
opt1_ts <- opt_design(TwoStageDesign(opt1_gs$design))
opt1_os <- opt_design(OneStageDesign(500, 2.0))
```

5.2.5 Test Cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(list(opt1_ts, opt1_gs, opt1_os),
                function(x) x$nloptr_return$iterations)

print(iters)
```

```
## [1] 2131  915   20
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Type one error rate constraint is tested for the three designs.

```
tmp     <- sapply(list(opt1_ts, opt1_gs, opt1_os),
                  function(x) sim_pr_reject(x$design, .0, datadist))
df_toer <- data.frame(
    toer = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_toer$toer <= alpha * (1 + tol)))

df_toer
```

```
##       toer           se
## 1 0.024975 0.0001560489
## 2 0.024978 0.0001560581
## 3 0.025116 0.0001564775
```

The power constraint can also be tested via simulation.

```
tmp     <- sapply(list(opt1_ts, opt1_gs, opt1_os),
                  function(x) sim_pr_reject(x$design, .2, datadist))
df_pow  <- data.frame(
    pow  = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
```

```
rm(tmp)

testthat::expect_true(all(df_pow$pow >= min_power * (1 - tol)))

df_pow
```

```
##        pow           se
## 1 0.799800 0.0004001501
## 2 0.799669 0.0004002482
## 3 0.799317 0.0004005115
```

The expected sample sizes should be ordered in a specific way.

```
testthat::expect_gte(
    evaluate(ess, opt1_os$design),
    evaluate(ess, opt1_gs$design)
)

testthat::expect_gte(
    evaluate(ess, init_design_gs),
    evaluate(ess, opt1_gs$design)
)

testthat::expect_gte(
    evaluate(ess, opt1_gs$design),
    evaluate(ess, opt1_ts$design)
)
```

The expected sample size of the optimal designs is simulated and compared to the outomce of `adoptr::evaluate()`.

```
ess_0 <- ExpectedSampleSize(datadist, H_0)

expect_equal(
    sim_n(opt1_os$design, .0, datadist)$n,
    evaluate(ess_0, opt1_os$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt1_gs$design, .0, datadist)$n,
    evaluate(ess_0, opt1_gs$design),
    tolerance = tol_n,
    scale = 1
)
```

```
expect_equal(
    sim_n(opt1_ts$design, .0, datadist)$n,
    evaluate(ess_0, opt1_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

Additionally, the sample sizes under the point prior are compared.

```
expect_equal(
    sim_n(opt1_os$design, .2, datadist)$n,
    evaluate(ess, opt1_os$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt1_gs$design, .2, datadist)$n,
    evaluate(ess, opt1_gs$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt1_ts$design, .2, datadist)$n,
    evaluate(ess, opt1_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

The n_2 function of the optimal two-stage design is expected to be monotonously decreasing.

```
testthat::expect_equal(
    sign(diff(opt1_ts$design@n2_pivots)),
    rep(-1, (order - 1))
)
```

5.3 Variant IV-2: Increase Power

5.3.1 Objective

The objective remains the same as before.

5.3.2 Constraints

The minimal required power is increased to 90%.

```
pow_cnstr_2 <- Power(datadist, prior) >= .9
```

5.3.3 Initial Design

The initial design is updated to a group-sequential design that fulfills the new power constraint.

```
order <- 5L

init_design_2_gs <- rpact_design(0.2, 0.025, 0.9, TRUE, order)

init_design_2    <- TwoStageDesign(init_design_2_gs)
```

5.3.4 Optimization

The optimal two-stage design is computed.

```
opt_design <- function(initial_design) {
    minimize(
        ess,
        subject_to(
            toer_cnstr,
            pow_cnstr_2
        ),
        initial_design = initial_design,
        opts = opts
    )
}

opt2_ts <- opt_design(init_design_2)
opt2_gs <- opt_design(init_design_2_gs)
opt2_os <- opt_design(OneStageDesign(500, 2.0))
```

5.3.5 Test Cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(list(opt2_ts, opt2_gs, opt2_os),
                function(x) x$nloptr_return$iterations)
```

```
print(iters)
```

```
## [1] 2988 1349   30
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Type one error rate constraint is tested for the three designs.

```
tmp     <- sapply(list(opt2_ts, opt2_gs, opt2_os),
                  function(x) sim_pr_reject(x$design, .0, datadist))
df_toer <- data.frame(
    toer = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_toer$toer <= alpha * (1 + tol)))

df_toer
```

```
##       toer           se
## 1 0.024980 0.0001560642
## 2 0.024946 0.0001559606
## 3 0.025116 0.0001564775
```

The power constraint can also be tested via simulation.

```
tmp     <- sapply(list(opt2_ts, opt2_gs, opt2_os),
                  function(x) sim_pr_reject(x$design, .2, datadist))
df_pow  <- data.frame(
    pow  = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_pow$pow >= .9 * (1 - tol)))

df_pow
```

```
##        pow           se
## 1 0.900131 0.0002998254
## 2 0.899828 0.0003002293
## 3 0.899523 0.0003006351
```

The expected sample sizes should be ordered in a specific way.

```
testthat::expect_gte(
    evaluate(ess, opt2_os$design),
```

```
    evaluate(ess, opt2_gs$design)
)

testthat::expect_gte(
    evaluate(ess, init_design_2_gs),
    evaluate(ess, opt2_gs$design)
)

testthat::expect_gte(
    evaluate(ess, opt2_gs$design),
    evaluate(ess, opt2_ts$design)
)
```

The expected sample size of the optimal designs is simulated and compared to the outomce of `adoptr::evaluate()`.

```
ess_0 <- ExpectedSampleSize(datadist, H_0)

expect_equal(
    sim_n(opt2_os$design, .0, datadist)$n,
    evaluate(ess_0, opt2_os$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt2_gs$design, .0, datadist)$n,
    evaluate(ess_0, opt2_gs$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt2_ts$design, .0, datadist)$n,
    evaluate(ess_0, opt2_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

Additionally, the sample sizes under the point prior are compared.

```
expect_equal(
    sim_n(opt2_os$design, .2, datadist)$n,
    evaluate(ess, opt2_os$design),
    tolerance = tol_n,
    scale = 1
)
```

```
expect_equal(
    sim_n(opt2_gs$design, .2, datadist)$n,
    evaluate(ess, opt2_gs$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt2_ts$design, .2, datadist)$n,
    evaluate(ess, opt2_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

The n_2 function of the optimal two-stage design is expected to be monotonously decreasing.

```
testthat::expect_equal(
    sign(diff(opt2_ts$design@n2_pivots)),
    rep(-1, (order - 1))
)
```

5.4 Variant IV-3: Increase Type One Error rate

5.4.1 Objective

Expected sample size under the point prior is minimized and has already been defined.

5.4.2 Constraints

The maximal type one error rate is increased to 5%.

```
toer_cnstr_2 <- Power(datadist, H_0) <= .05
```

5.4.3 Initial Design

The initial design is updated to a group-sequential design that fulfills the new type one error rate constraint.

```
order <- 5L

init_design_3_gs <- rpact_design(0.2, 0.05, 0.9, TRUE, order)
```

```
init_design_3    <- TwoStageDesign(init_design_3_gs)
```

5.4.4 Optimization

The optimal two-stage design is computed.

```
opt_design <- function(initial_design) {
    minimize(
        ess,
        subject_to(
            toer_cnstr_2,
            pow_cnstr_2
        ),
        initial_design = initial_design,
        opts = opts
    )
}

opt3_ts <- opt_design(init_design_3)
opt3_gs <- opt_design(init_design_3_gs)
opt3_os <- opt_design(OneStageDesign(500, 2.0))
```

5.4.5 Test Cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(list(opt3_ts, opt3_gs, opt3_os),
                function(x) x$nloptr_return$iterations)

print(iters)
```

```
## [1] 2833 1124   27
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Type one error rate constraint is tested for the three designs.

```
tmp     <- sapply(list(opt3_ts, opt3_gs, opt3_os),
                  function(x) sim_pr_reject(x$design, .0, datadist))
df_toer <- data.frame(
    toer = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)
```

```
testthat::expect_true(all(df_toer$toer <= .05 * (1 + tol)))

df_toer
```

```
##       toer           se
## 1 0.050175 0.0002183060
## 2 0.049980 0.0002179038
## 3 0.050150 0.0002182545
```

The power constraint can also be tested via simulation.

```
tmp     <- sapply(list(opt3_ts, opt3_gs, opt3_os),
                  function(x) sim_pr_reject(x$design, .2, datadist))
df_pow  <- data.frame(
    pow  = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_pow$pow >= .9 * (1 - tol)))

df_pow
```

```
##        pow           se
## 1 0.900059 0.0002999215
## 2 0.900317 0.0002995770
## 3 0.899606 0.0003005248
```

The expected sample sizes should be ordered in a specific way.

```
testthat::expect_gte(
    evaluate(ess, opt3_os$design),
    evaluate(ess, opt3_gs$design)
)

testthat::expect_gte(
    evaluate(ess, init_design_3_gs),
    evaluate(ess, opt3_gs$design)
)

testthat::expect_gte(
    evaluate(ess, opt3_gs$design),
    evaluate(ess, opt3_ts$design)
)
```

The expected sample size of the optimal designs is simulated and compared to the outomce of `adoptr::evaluate()`.

```
ess_0 <- ExpectedSampleSize(datadist, H_0)

expect_equal(
    sim_n(opt3_os$design, .0, datadist)$n,
    evaluate(ess_0, opt3_os$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt3_gs$design, .0, datadist)$n,
    evaluate(ess_0, opt3_gs$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt3_ts$design, .0, datadist)$n,
    evaluate(ess_0, opt3_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

Additionally, the sample sizes under the point prior are compared.

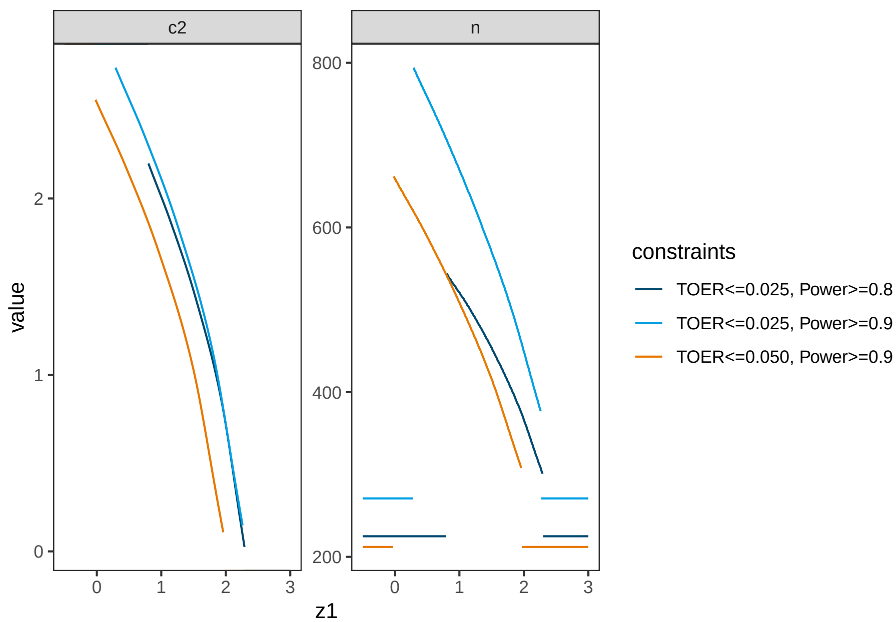
```
expect_equal(
    sim_n(opt3_os$design, .2, datadist)$n,
    evaluate(ess, opt3_os$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt3_gs$design, .2, datadist)$n,
    evaluate(ess, opt3_gs$design),
    tolerance = tol_n,
    scale = 1
)

expect_equal(
    sim_n(opt3_ts$design, .2, datadist)$n,
    evaluate(ess, opt3_ts$design),
    tolerance = tol_n,
    scale = 1
)
```

The n_2 function of the optimal two-stage design is expected to be monotonously decreasing.

```
testthat::expect_equal(
    sign(diff(opt3_ts$design@n2_pivots)),
    rep(-1, (order - 1))
)
```

5.5 Plot Two-Stage Designs

The optimal two-stage designs stemming from the three different variants are plotted together.



Chapter 6

Scenario V: Single-arm design, medium effect size

6.1 Details

In this scenario, a classical two-arm trial with normal test statistic and known variance (w.l.o.g. variance of the test statistic is 1). This situation corresponds to a classical z-test for a difference in population means.

```
datadist <- Normal(two_armed = TRUE)
```

The null hypothesis is no population mean difference, i.e. $\mathcal{H}_0 : \delta \leq 0$.

```
H_0 <- PointMassPrior(.0, 1)
```

An alternative effect size of $\delta = 0.3$ with point prior distribution is assumed.

```
prior <- PointMassPrior(.3, 1)
```

Across all variants in this scenario, the one-sided maximal type one error rate is restricted to

```
alpha <- 0.025
```

and the power at the point alternative of $\delta = 0.3$ must be at least

```
min_power <- 0.8
```

I.e. throughout this sceanrio, we always use the two constraints

```
toer_cnstr <- Power(datadist, H_0) <= alpha
```

and

```
pow_cnstr <- Power(datadist, prior) >= min_power
```

6.2 Variant V-1, sensitivity to integration order

6.2.1 Objective

Expected sample size under the respective prior is minimized, i.e., $\boldsymbol{E}\big[n(\mathcal{D})\big]$.

```
ess <- ExpectedSampleSize(datadist, prior)
```

6.2.2 Constraints

No additional constraints are considered in this variant.

6.2.3 Initial Design

A fixed design for these parameters would require 176 subjects per group. We use the half of this as initial values for the sample sizes. The initial stop for futility is at $c_1^f = 0$, i.e., if the effect shows in the opponent direction to the alternative. The starting values for the efficacy stop and for c_2 is the $1 - \alpha$-quantile of the normal distribution.

```
init_design <- function(order) {
    TwoStageDesign(
        n1 = ceiling(pwr::pwr.t.test(d = .3,
                                     sig.level = .025,
                                     power = .8,
                                     alternative = "greater")$n) / 2,
        c1f = 0,
        c1e = qnorm( 1 - 0.025),
        n2 = ceiling(pwr::pwr.t.test(d = .3,
                                     sig.level = .025,
                                     power = .8,
                                     alternative = "greater")$n) / 2,
        c2 = qnorm(1 - 0.025),
        order = order
)
}
```

6.2.4 Optimization

The optimal design is computed for three different integration orders: 5, 8, and 11.

```
opt_design <- function(order) {
    minimize(
        ess,
        subject_to(
            toer_cnstr,
            pow_cnstr
        ),
        initial_design = init_design(order),
        opts = opts
    )
}

opt1 <- lapply(c(5, 8, 11), function(x) opt_design(x))
```

6.2.5 Test cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(opt1, function(x) x$nloptr_return$iterations)

print(iters)
```

```
## [1] 2328 4226 8913
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Check type one error rate control.

```
tmp     <- sapply(opt1, function(x) sim_pr_reject(x$design, .0, datadist))
df_toer <- data.frame(
    toer = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_toer$toer <= alpha * (1 + tol)))

df_toer
```

```
##       toer           se
## 1 0.024975 0.0001560489
## 2 0.024956 0.0001559911
```

```
## 3 0.024950 0.0001559728
```

Check the power constraint.

```
tmp     <- sapply(opt1, function(x) sim_pr_reject(x$design, .3, datadist))
df_pow  <- data.frame(
    power = as.numeric(tmp[1, ]),
    se    = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_pow$pow >= min_power * (1 - tol)))

df_pow
```

```
##      power           se
## 1 0.799791 0.0004001569
## 2 0.799696 0.0004002280
## 3 0.799678 0.0004002415
```

Check expected sample size under the prior.

```
tmp    <- sapply(opt1, function(x) sim_n(x$design, .3, datadist))
df_ess <- data.frame(
    n  = as.numeric(tmp[1, ]),
    se = as.numeric(tmp[2, ])
)
rm(tmp)

df_ess
```

```
##          n         se
## 1 141.9614 0.04874384
## 2 141.9801 0.04875722
## 3 141.9822 0.04875670
```

6.3 Variant V-2, utility maximization

6.3.1 Objective

In this case, a utility function consisting of expected sample size and power is minimized.

```
pow <- Power(datadist, prior)
ess <- ExpectedSampleSize(datadist, prior)

obj <- function(lambda) {
```

```
  composite({ess - lambda * pow})
}
```

6.3.2 Constraints

The type one error rate is controlled at 0.025 on the boundary of the null hypothesis. Hence, the previous inequality can still be used. There is no constraint on power anymore because power is part of the objective utility function.

6.3.3 Initial Design

The previous initial design with order 5 is applied.

6.3.4 Optimization

The optimal design is computed for two values of λ: 200 and 500.

```
opt2_design <- function(lambda) {

    minimize(
        obj(lambda),
        subject_to(
            toer_cnstr
        ),
        initial_design = init_design(5),
        opts = opts
    )
}

opt2 <- lapply(c(200, 500), function(x) opt2_design(x))
```

6.3.5 Test cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(opt2, function(x) x$nloptr_return$iterations)

print(iters)
```

```
## [1]  2062 13606
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Check type one error rate control for both designs via simulation.

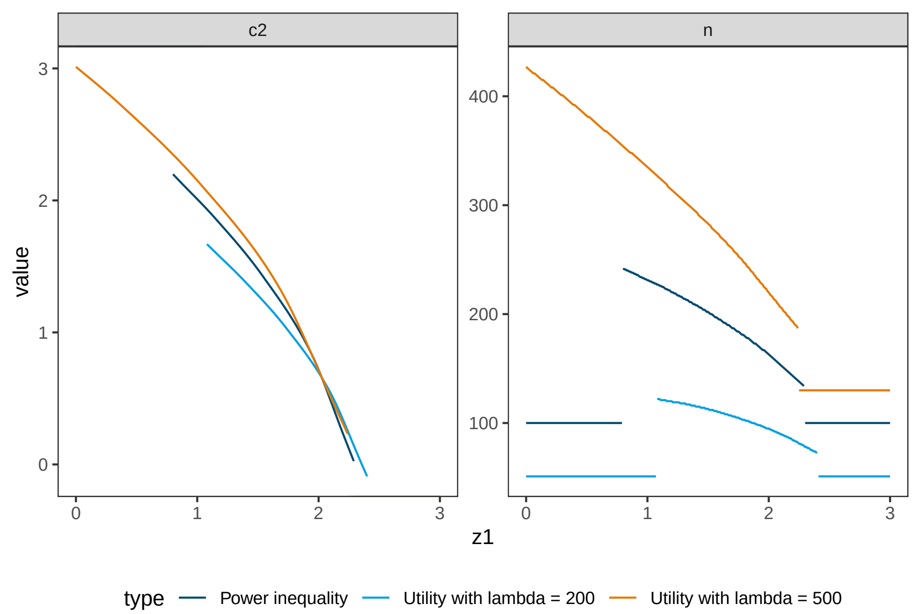
```
tmp     <- sapply(opt2, function(x) sim_pr_reject(x$design, 0, datadist))
df_toer <- data.frame(
    toer = as.numeric(tmp[1, ]),
    se   = as.numeric(tmp[2, ])
)
rm(tmp)

testthat::expect_true(all(df_toer$toer <= alpha * (1 + tol)))

df_toer
```

```
##        toer           se
## 1 0.025024 0.0001561980
## 2 0.024983 0.0001560733
```

Check if the power of the design with higher λ is larger.

```
testthat::expect_gte(
    evaluate(pow, opt2[[2]]$design),
    evaluate(pow, opt2[[1]]$design)
)
```

Finally the three designs computed so far are plotted together to allow comparison.



6.4 Variant V-3, n1-penalty

In this case, the influence of the regularization term `N1()` is investigated.

6.4.1 Objective

In this case, a mixed criterion consisting of expected sample size and n_1 is minimized.

```
N1 <- N1()

obj3 <- function(lambda) {
  composite({ess + lambda * N1})
}
```

6.4.2 Constraints

The inequalities from variant V.1 can still be used.

6.4.3 Initial Design

The previous initial design with order 5 is applied.

6.4.4 Optimization

The optimal design is computed for two values of λ: 0.05 and 0.2.

```
opt3_design <- function(lambda) {

    minimize(
        obj3(lambda),
        subject_to(
            toer_cnstr,
            pow_cnstr
        ),
        initial_design = init_design(5),
        opts = opts
    )
}

opt3 <- lapply(c(.05, .2), function(x) opt3_design(x))
```

6.4.5 Test cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(opt3, function(x) x$nloptr_return$iterations)

print(iters)
```

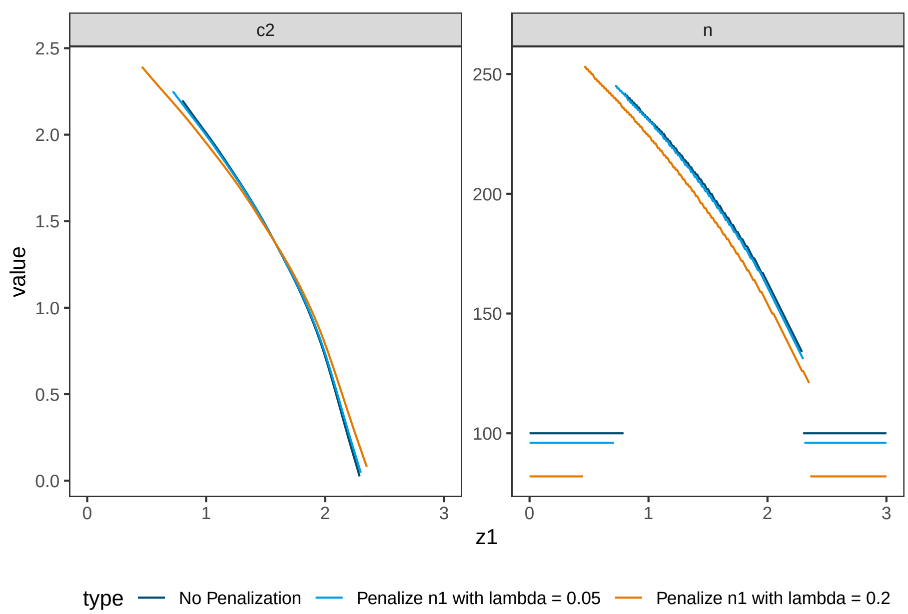
```
## [1] 2233 2478
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Check if the n1 regularizer of the design with higher λ is lower.

```
testthat::expect_lte(
    evaluate(N1, opt3[[2]]$design),
    evaluate(N1, opt3[[1]]$design)
)


testthat::expect_lte(
    evaluate(N1, opt3[[1]]$design),
    evaluate(N1, opt1[[1]]$design)
)
```

Finally the three designs computed so far are plotted together to allow comparison.



6.5 Variant V-4, n2-penalty

In this case the average over n_2 is penalized by the predefined score `AverageN2`.

6.5.1 Objective

In this case, a mixed criterion consisting of expected sample size and average of n_2 is minimized.

```
avn2 <- AverageN2()

obj4 <- function(lambda) {
  composite({ess + lambda * avn2})
}
```

6.5.2 Constraints

The inequalities from variant V.1 can still be used.

6.5.3 Initial Design

The previous initial design with order 5 is applied.

6.5.4 Optimization

The optimal design is computed for two values of λ: 0.01 and 0.1.

```
opt4_design <- function(lambda) {
    minimize(
        obj4(lambda),
        subject_to(
            toer_cnstr,
            pow_cnstr
        ),
        initial_design = init_design(5),
        upper_boundary_design = get_upper_boundary_design(init_design(5), c2_buffer=3),
        opts = opts
    )
}

opt4 <- lapply(c(.01, .1), function(x) opt4_design(x))
```

6.5.5 Test cases

Check if the optimization algorithm converged in all cases.

```
iters <- sapply(opt4, function(x) x$nloptr_return$iterations)

print(iters)
```

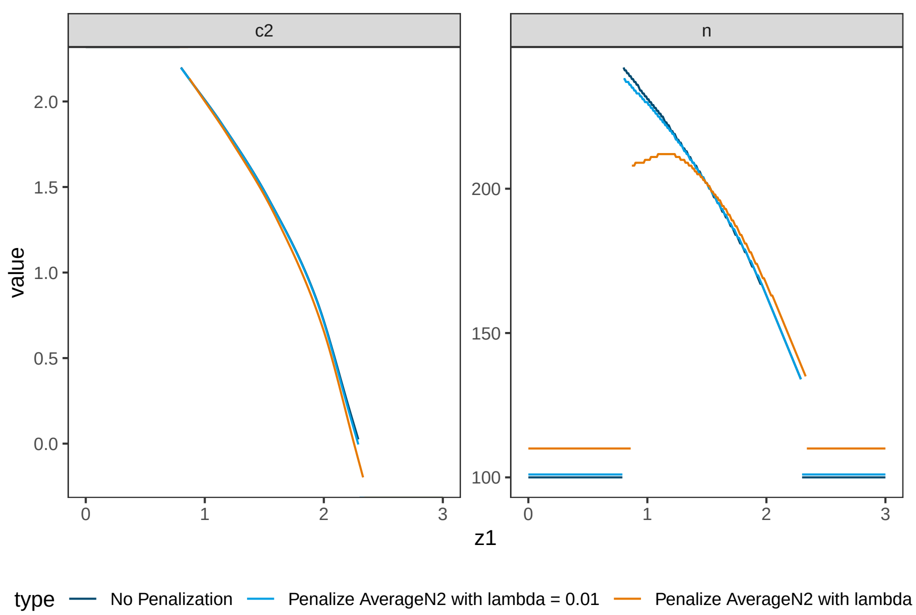
```
## [1] 2196 2376
```

```
testthat::expect_true(all(iters < opts$maxeval))
```

Check if the average n_2 regularizer of the design with higher λ is lower.

```
testthat::expect_lte(
    evaluate(avn2, opt4[[2]]$design),
    evaluate(avn2, opt4[[1]]$design)
)


testthat::expect_lte(
    evaluate(avn2, opt4[[1]]$design),
    evaluate(avn2, opt1[[1]]$design)
)
```

Finally the three designs computed so far are plotted together to allow comparison.



Bibliography

Bauer, P., Bretz, F., Dragalin, V., König, F., and Wassmer, G. (2015). Twenty-five years of confirmatory adaptive designs: opportunities and pitfalls. *Statistics in Medicine*, 35(3):325–347.

Pilz, M., Kunzmann, K., Herrmann, C., Rauch, G., and Kieser, M. (2019). A variational approach to optimal two-stage designs. *Statistics in Medicine*, pages 1–13.

Wassmer, G. and Brannath, W. (2016). *Group sequential and confirmatory adaptive designs in clinical trials*. Springer Series in Pharmaceutical Statistics -. Springer International Publishing.

Wassmer, G. and Pahlke, F. (2018). *rpact: Confirmatory Adaptive Clinical Trial Design and Analysis*. R package version 1.0.0.

Wickham, H., Studio, R., and Team, R. C. (2018). *testthat: Unit Testing for R*. R package version 2.0.1.