Validation Report for drugdevelopR

Johannes Cepicka, Lukas D. Sauer

2024-12-05

Contents

General introduction and validation plan **2**

Release details **3**
Package Information . . . 3
Authors . . . 3
Traceability . . . 5

Risk Assessment **8**

User Requirement Specification **10**
01. Basic setting . . . 10
02. Bias adjustment – Discounting phase II results . . . 11
03. Multitrial – Programs with several phase III trials . . . 13
04. Multiarm – Programs with more than one experimental arm . . . 14
05. Multiple endpoints . . . 15

Test Cases **17**
01. Basic setting test cases . . . 17
02. Bias adjustment test cases . . . 21
03. Multitrial test cases . . . 25
04. Multiarm test cases . . . 29
05. Multiple endpoints test cases . . . 31

Test Results **34**

References **45**

General introduction and validation plan

In the planning of confirmatory studies, the determination of the sample size is essential, as it has a significant impact on the chances of achieving the study objective. Based on the work of Götte et al. (Götte et al. 2015), methods for optimal sample size and go/no-go decision rules for the example of phase II/III drug development programs within a utility-based, Bayesian-frequentist framework were developed, which was improved by Preussler in her dissertation "Integrated Planning of Pilot and Subsequent Confirmatory Study in Clinical Research - Finding Optimal Designs in a Utility-Based Framework" (Preussler 2020) to include further extensions. Additionally, methods for multiple endpoints were implemented (Kieser et al. 2018). In order to facilitate the practical application of these approaches, R Shiny applications for a basic setting as well as for three extensions were implemented. The extension of this project to a fully functional software product was funded by the German Research Organisation DFG within a project entitled "Integrated Planning of Drug Development Programs - drugdevelopR". The package was developed at the Institute of Medical Biometry (IMBI) in Heidelberg, Germany.

In order to assure unchanging scientific quality, in order to document that the package's user requirements were met and in order to provide evidence of the package's validity to the general public, we supply the following human-readable validation report. Within this report, we present the results of a validation suite developed at IMBI specifically for this package. The validation suite comprises several benchmark/testing scenarios of optimal drug development planning as presented in published work with scientific quality assurance. Published results are compared with the results of the software. The validation plan aims for full coverage of all package methods.

The validation suite was programmed using the *valtools* R package (Hughes et al. 2021). We closely followed the R package validation framework (Hughes 2021) of the PHUSE Working Group on "Data Visualisation & Open Source Technology".

With the aim of avoiding confusion, it should be noted that the framework makes a clear distinction between testing (unit testing) and validating a package. Testing on the one hand means checking the program from the software developer's perspective. The so-called unit tests usually cover small sections of the program, each checking one specific part of the software, e.g. a function. With their help, the programmer assures that the code works as intended. Unit tests aim for a code coverage close to one-hundred percent. Validation on the other hand means checking the program from the end user's point of view. By checking larger parts of software within one test case, the validation process provides evidence that the program will deliver the expected results in a production environment.

Both unit tests and a validation suite were implemented for the drugdevelopR package. However, this report only covers validation. Whenever we refer to test cases or test code within this document, we mean tests for validation and not unit tests.

In the following, we supply a brief introduction to this validation framework. The framework comprises four distinct steps:

1. Requirements: Programmers, subject matter experts and end users formulate clear, general expectations of the program's functionality. (In our case, subject matter experts and end users coincide. Hence, the requirements were discussed between the programmer and two experts.) The requirements are readable by any expert without any programming experience. Neither program code nor function names are defined within this part of the validation framework. Risk assessments for the likelihood of errors within each requirement and for the impact of possible errors are documented for each requirement.
2. Test cases: For each of the requirements, the programmer writes one or several test cases. These are concise plain-text descriptions of how to verify that the requirement has been met by the package. It clearly specifies program input, the function names of the functions to be used, and expected program output. However, no program code is supplied. Each test case usually covers a use case that could be expected in a real-life application of the program.
3. Test code: Another programmer, who is not involved in the package code development, will then implement the test cases in R. The test code is clearly structured and thereby demonstrates that it follows the corresponding test case. The external programmer writes the test code solely using the

description of the test cases and the package's documentation, without deeper insight into the package's code. This has the advantage that misunderstandings, poor documentation and other pitfalls will be discovered by an independent user before software release.

4. Validation report: As the last step, the primary programmer generates a human-readable validation report. Within this report, requirements and test cases will be listed and the results of the test code are presented. Thus, the whole validation process and its outcome are available to anyone who wants to verify the package's validity. After changes to the program, the validation report can be easily generated (possibly after adding additional test cases for new functionality).

Release details

Package Information

Change Log

Version	Effective Date	Activity Description
0.1.0	2022-08-21	Validation release notes for version 0.1.0

Validation Environment

Type	Resource	Version Detail
system	OS	Windows 11 x64 (build 26100)
	R	4.4.2 (2024-10-31
package_req	cubature	2.1.1
	doParallel	1.0.17
	foreach	1.5.2
	iterators	1.0.14
	MASS	7.3.61
	msm	1.8.2
	mvtnorm	1.3.2
	parallel	4.4.2
	progressr	0.15.1
	stats	4.4.2
extended_req	covr	3.6.4
	devtools	2.4.5
	kableExtra	1.4.0
	knitr	1.49
	magrittr	2.0.3
	rmarkdown	2.29
	testthat	3.2.1.1
session	usethis	3.1.0
	valtools	0.4.0

Authors

Requirements

Requirement ID	Editor	Edit Date
01. Basic setting	Johannes Cepicka	2022-08-16
02. Bias adjustment	Johannes Cepicka	2022-08-16
03. Multitrial	Johannes Cepicka	2022-08-16
04. Multiarm	Johannes Cepicka	2022-08-16
05. Multiple endpoints	Johannes Cepicka	2022-08-16

Test Case Authors

Test Case ID	Editor	Edit Date
01. Basic setting test cases	Johannes Cepicka	2022-08-16
02. Bias adjustment test cases	Johannes Cepicka	2022-08-16
03. Multitrial test cases	Johannes Cepicka	2022-08-16
04. Multiarm test cases	Johannes Cepicka	2022-08-16
05. Multiple endpoints test cases	Johannes Cepicka	2022-08-16

Test Code Authors

Test Code ID	Editor	Edit Date
01.01		2022-09-13
01.02		2022-10-10
01.03		
01.04		
01.05		
01.06		
01.07		2022-10-11
01.08		2022-10-12
01.09		
01.10		2022-10-11
01.11		
01.12		
01.13		2023-03-30
01.14		
01.15		
02.01		2022-12-07
02.02		
02.03		
02.04		
02.05		2022-12-13
02.06		2022-12-14
02.07		
02.09		
02.10		
02.11		2022-12-20
02.12		
03.01		
03.02		
03.03		
03.04		
03.05		
03.06		
03.07		
03.08		
03.09		
03.10		
03.11		
03.12		
03.13		
03.14		
04.01		

04.02		
04.03		
04.04		
04.05		
04.06		
04.07		
04.08		
04.09		
04.10		
05.01		
05.02		
05.03		2022-12-29
05.04		
05.05		
05.06		
05.07		
05.08		
05.09		
05.10		

Traceability

Requirement Name	Requirement ID	Test Case Name	Test Cases
Requirement 01	01.03	01. Basic setting test cases	01.01
	01.06		01.01
	01.12		01.01
	01.15		01.01
	01.03		01.02
	01.05		01.02
	01.14		01.02
	01.16		01.02
	01.10		01.03
	01.12		01.03
	01.09		01.04
	01.13		01.04
	01.11		01.05
	01.14		01.05
	01.15		01.05
	01.04		01.06
	01.07		01.07
	01.02		01.08
	01.05		01.08
	01.12		01.08
	01.02		01.09
	01.06		01.09
	01.01		01.10
	01.06		01.10
	01.01		01.11
	01.05		01.11
	01.15		01.11
	01.08		01.12
	01.01		01.13
	01.13		01.13
	01.14		01.13
	01.02		01.14
	01.13		01.14
	01.14		01.14
	01.03		01.15
	01.13		01.15
	01.14		01.15
	02.03		02.01
	02.05		02.01
	02.10		02.01
	02.19		02.01
	02.03		02.02
	02.05		02.02
	02.11		02.02
	02.17		02.02

Requirement 02	02.06	02. Bias adjustment test cases	02.03
	02.12		02.03
	02.16		02.03
	02.20		02.03
	02.13		02.04
	02.20		02.04
	02.04		02.05
	02.07		02.05
	02.15		02.05
	02.08		02.06
	02.18		02.06
	02.14		02.07
	02.01		02.09
	02.04		02.09
	02.11		02.09
	02.01		02.10
	02.05		02.10
	02.13		02.10
	02.02		02.11
	02.04		02.11
	02.10		02.11
	02.02		02.12
	02.05		02.12
	02.12		02.12
Requirement 03	03.03	03. Multitrial test cases	03.01
	03.05		03.01
	03.11		03.01
	03.20		03.01
	03.03		03.02
	03.10		03.02
	03.15		03.02
	03.18		03.02
	03.03		03.03
	03.07		03.03
	03.16		03.03
	03.12		03.04
	03.04		03.05
	03.14		03.05
	03.20		03.05
	03.21		03.05
	03.13		03.06
	03.20		03.06
	03.02		03.07
	03.05		03.07
	03.11		03.07
	03.02		03.08
	03.14		03.08
	03.02		03.09
	03.04		03.09
	03.21		03.09
	03.06		03.10
	03.10		03.10
	03.17		03.10
	03.01		03.11
	03.05		03.11
	03.11		03.11
	03.15		03.11
	03.09		03.12
	03.01		03.13
	03.08		03.13
	03.19		03.13
	03.01		03.14
	03.04		03.14
	03.10		03.14
	04.03		04.01
	04.08		04.01
	04.12		04.01
	04.03		04.02
	04.09		04.02
	04.12		04.02
	04.03		04.03
	04.04		04.03
	04.14		04.03
	04.07		04.04
	04.02		04.05
	04.08		04.05
	04.14		04.05
	04.02		04.06
	04.09		04.06
	04.14		04.06
	04.02		04.07

	04.06		04.07
	04.11		04.07
	04.01		04.08
	04.08		04.08
	04.13		04.08
	04.01		04.09
	04.09		04.09
	04.13		04.09
	04.01		04.10
	04.05		04.10
	04.10		04.10
Requirement 05	05.01	05. Multiple endpoints test cases	05.01
	05.04		05.01
	05.11		05.01
	05.15		05.01
	05.01		05.02
	05.04		05.02
	05.05		05.02
	05.10		05.02
	05.01		05.03
	05.03		05.03
	05.06		05.03
	05.09		05.03
	05.01		05.04
	05.03		05.04
	05.07		05.04
	05.12		05.04
	05.08		05.05
	05.02		05.06
	05.04		05.06
	05.05		05.06
	05.10		05.06
	05.02		05.07
	05.04		05.07
	05.08		05.07
	05.02		05.08
	05.03		05.08
	05.06		05.08
	05.09		05.08
	05.02		05.09
	05.03		05.09
	05.07		05.09
	05.12		05.09
	05.11		05.10
	05.13		05.10
	05.14		05.10

Risk Assessment

Requirement Name	Requirement ID	Risk Assessment
01. Basic setting	01.01	Low Risk, Very High Impact
	01.02	Low Risk, Very High Impact
	01.03	Low Risk, Very High Impact
	01.04	Medium Risk, Low Impact
	01.05	Low Risk, High Impact
	01.06	Medium Risk, High Impact
	01.07	Low Risk, Low Impact
	01.08	High Risk, Low Impact
	01.09	Low Risk, High Impact
	01.10	Low Risk, High Impact
	01.11	Low Risk, High Impact
	01.12	Low Risk, High Impact
	01.13	Low Risk, Medium Impact
	01.14	Low Risk, Medium Impact
	01.15	Low Risk, Medium Impact
	01.16	Low Risk, Medium Impact
02. Bias adjustment	02.01	Medium Risk, High Impact
	02.02	Medium Risk, High Impact
	02.03	Low Risk, High Impact
	02.04	Medium Risk, High Impact
	02.05	High Risk, High Impact
	02.06	Medium Risk, High Impact
	02.07	Medium Risk, High Impact
	02.08	Medium Risk, High Impact
	02.09	High Risk, Low Impact
	02.10	Medium Risk, High Impact
	02.11	Medium Risk, High Impact
	02.12	Medium Risk, High Impact
	02.13	Medium Risk, High Impact
	02.14	Medium Risk, Medium Impact
	02.15	Medium Risk, Medium Impact
	02.16	Low Risk, Medium Impact
	02.17	Low Risk, Medium Impact
	02.18	Low Risk, Medium Impact
	02.19	Low Risk, Medium Impact
	02.20	Low Risk, Medium Impact
	03.01	Medium Risk, High Impact
	03.02	Medium Risk, High Impact
	03.03	Low Risk, High Impact
	03.04	Medium Risk, High Impact
	03.05	High Risk, High Impact
	03.06	Low Risk, High Impact
	03.07	Low Risk, High Impact
	03.08	Low Risk, High Impact
	03.09	High Risk, Low Impact
	03.10	Medium Risk, High Impact
	03.11	Medium Risk, High Impact
	03.12	High Risk, Low Impact
	03.13	Low Risk, Low Impact
	03.14	Very High Risk, Medium Impact
	03.15	Medium Risk, Medium Impact

	03.16	Low Risk, Medium Impact
	03.17	Low Risk, Medium Impact
	03.18	Low Risk, Medium Impact
	03.19	Low Risk, Medium Impact
	03.20	Low Risk, Medium Impact
	03.21	Low Risk, Medium Impact
04. Multiarm	04.01	Medium Risk, High Impact
	04.02	Medium Risk, High Impact
	04.03	Low Risk, High Impact
	04.04	Low Risk, High Impact
	04.05	Low Risk, High Impact
	04.06	Low Risk, High Impact
	04.07	High Risk, Low Impact
	04.08	Medium Risk, High Impact
	04.09	Medium Risk, High Impact
	04.10	Low Risk, Medium Impact
	04.11	Low Risk, Medium Impact
	04.12	Low Risk, Medium Impact
	04.13	Low Risk, Medium Impact
	04.14	Low Risk, Medium Impact
05. Multiple endpoints	05.01	High Risk, High Impact
	05.02	High Risk, High Impact
	05.03	High Risk, High Impact
	05.04	Very High Risk, High Impact
	05.05	Low Risk, High Impact
	05.06	Low Risk, High Impact
	05.07	Low Risk, High Impact
	05.08	High Risk, Low Impact
	05.09	Low Risk, Medium Impact
	05.10	Low Risk, Medium Impact
	05.11	Low Risk, Medium Impact
	05.12	Low Risk, Medium Impact
	05.13	High Risk, High Impact
	05.14	High Risk, High Impact
	05.15	High Risk, High Impact

User Requirement Specification

In the following section, we will specify functionality that the end user can expect from the *drugdevelopR* package. We will use the following terms in the text, following the definitions from (Preussler 2020):

- In the most basic setting, a *drug development program* consists of a single exploratory phase II trial which is, in case of promising results, followed by one confirmatory phase III trial testing the superiority of an experimental treatment over a control treatment. The phase II trial and the phase III trial are two-arm randomized studies, each with balanced sample size allocation. They are performed independently with the same primary endpoint. Both trials draw from the same population. More complex drug development programs will be defined throughout the text.
- A drug development program is called a *successful program* if it proceeds from phase II to phase III and if there is a statistically significant positive treatment effect in phase III of the program.
- The *utility function* of a drug development program calculates the difference between costs and gains of the program. The costs are composed of fixed costs and variable costs which depend on the number of participants in the program. The gains are given as the expected revenue of a successful program and depend on the size of the treatment effect. The idea is that treatments with bigger effect yield bigger revenue on the market. To this end, the user will be asked to specify three effect size categories as well as expected gains for each category.
- The *optimal sample size* of a drug development program is the number of participants in phase II that maximizes the utility function. The maximum of the utility function is calculated on the optimization set as defined by the user.

01. Basic setting

In the simplest case, the drug development program consists of a single exploratory phase II trial which is, in case of promising results, followed by one confirmatory phase III trial testing the superiority of an experimental treatment over a control treatment. The package should ask for the following user input values specific to the particular drug development program in consideration:

- Significance level,
- Assumed true treatment effects (for the intervention arm and the control arm),
- Event rates for phase II and III (for the time-to-event setting)
- Optimization set for number of participants in phase II,
- Optimization sets for go/no-go decision rule,
- Boundaries for effect size categories,
- Fixed and variable costs for each phase,
- Expected gains for each effect size,
- Cost constraint (maximum costs),
- Sample size constraint (maximum sample size),
- Constraint on the success probability (minimum expected probability of a successful program),
- A parameter to choose whether the true treatment effect is known ("fixed") or whether the treatment effect is unknown and follows a prior distribution as specified by the following parameters:
  - Amount of information for true treatment effects, i.e. a parameter for calculating the variance of the prior distribution: this is the sample size for the normal and the binary distribution and the number of events for the time-to-event setting
  - Weights for prior distributions,
  - Boundaries for truncation,
- Number of clusters for parallel computing.

These input values are necessary for all settings, not just for the basic setting. Input values which are only relevant in the basic setting are the following:

- A parameter γ that models an offset between the treatment effects in phase II and III: the treatment effect of phase III is then calculated by adding γ to the treatment effect of phase II
- A parameter to choose whether skipping phase II is an option.

Based on these input values, the program should calculate the optimal sample size and the optimal threshold value as well as the expected utility for this sample size and threshold value. As the program is required to calculate these two results in every sub-requirement within this section, we will refer to them as "the results" for the sake of simplicity. There are different possibilities how the outcome variables can be distributed. In order to fulfill the needs of different trials, the program should meet the following requirements concerning outcome variables:

- 01.01: Calculate the results in the basic setting for normally distributed outcome variables.
- 01.02: Calculate the results in the basic setting for binary outcome variables.
- 01.03: Calculate the results in the basic setting for time-to-event outcome variables.

The basic setting should be adaptable to more refined use cases. One special use case may be skipping the phase II, e.g. when enough information is available to proceed directly to a confirmatory trial. In this case, no optimization for phase II sample size and threshold value should be performed, but the sample size and utility of the phase III trial should still be calculated correctly. The treatment effect for the phase III planning should then be calculated as the median of the prior distribution. This leads to the following requirement:

- 01.04: Upon user selection, calculate the results for a setting where the phase II trial is skipped.

Several other options for the regular setting (including both phase II and phase II should be available):

- 01.05: Upon user selection, calculate the results for fixed treatment effects.
- 01.06: Upon user selection, calculate the results for treatment effects modeled on a user-specified prior distribution depending on the distribution of the outcome variable (normal, binary, time to event).
- 01.07: Upon user selection, calculate the results for custom population parameters (i.e. non-zero values of the offset parameter γ).
- 01.08: Upon user selection, the program should calculate the results using parallel computing, i.e. using more than one core. Finally, the program should be able user-specified constraints into account. In particular, the program should meet the following requirements concerning constraint violations:
- 01.09: Automatically set the internal utility function to -9999 for all sample sizes which exceed the user-defined maximum number of participants in phase II and III. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 01.10: Automatically set the internal utility function to -9999 for all sample sizes whose costs exceed the user-defined maximum cost limit. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 01.11: Automatically set the internal utility function to -9999 for all sample sizes whose drug development program success probability is lower than the user-defined miminum success probability. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.

In addition to the main results of optimal sample size, the optimal threshold value and the expected utility, the program should be able to return the following additional data concerning the drug development program:

- 01.12: Return the pre-specified constraint on the total costs as well as the actually calculated costs of phase II and III for the optimal sample size and threshold.
- 01.13: Return the total sample size as well as the separate sample sizes in phase II and III.
- 01.14: Return the probability that the program proceeds to phase III.
- 01.15: Return the probability of a successful program (for every effect size and in total).
- 01.16: Return the number of events in phase II and III (in the time-to-event-setting).

02. Bias adjustment – Discounting phase II results

As the drug development programs only continue to the next stage when preceding trials are successful, estimated treatment effects may be systematically too optimistic. The program should extend the basic setting by implementing bias adjustment . As in the basic setting, the drug development program consists of a single exploratory phase II trial which is, in case of a promising result, followed by one confirmatory

phase III trial. The same time-to-event, binary, or normally distributed endpoint is used in phase II and III, respectively. In addition to the general parameters specified in the section on the basic setting, the user should be able to provide the following additional parameters:

- A parameter to choose the bias adjustment method (additive or multiplicative adjustment),
- Parameters to determine the size of the adjustment.

As before, the program should correctly calculate the optimal sample size, the optimal threshold value and the corresponding expected utility taking the selected adjustment method and adjustment parameter as well as all other user input parameters into account. It should be adaptable to different use cases as before. Thus, we get the following requirements:

- 02.01: Calculate the results in the bias setting for normally distributed outcome variables.
- 02.02: Calculate the results in the bias setting for binary outcome variables.
- 02.03: Calculate the results in the bias setting for time-to-event outcome variables.
- 02.04: Calculate the results for fixed treatment effects.
- 02.05: Calculate the results for treatment effects modeled on a user-specified prior distribution.
- 02.06: Automatically set the internal utility function to -9999 for all sample sizes which exceed the user-defined maximum number of participants in phase II and III. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 02.07: Automatically set the internal utility function to -9999 for all sample sizes whose costs exceed the user-defined maximum cost limit. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 02.08: Automatically set the internal utility function to -9999 for all sample sizes whose drug development program success probability is lower than the user-defined miminum success probability. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 02.09: Upon user selection, the program should calculate the results using parallel computing, i.e. using more than one core.

We considered two different adjustment methods to discount (possibly) too optimistic phase II results, an additive method and a multiplicative one. Both methods adjust the estimate of the observed treatment effect of phase II. The user should be able to decide which method they want to use. If the user selects additive adjustment or multiplicative adjustment, the program should correctly adjust the treatment effect in accordance with the selected method. If the user selects the option “both”, the program should return the results for the two adjustment methods separately. If the user selects the option “all”, the program should return separate results for four different adjustment methods: the two adjustment methods named above as well as an additive and a multiplicative adjustment method that not only adjust the treatment effect but also the threshold value for the decision rule. The trivial adjustment parameters 0.5 or 1 should return the results from the basic setting. Therefore, we state the following requirements:

- 02.10: Upon user selection, calculate the results using the additive adjustment method (i.e. adapting the lower bound of the confidence interval).
- 02.11: Upon user selection, calculate the results using the multiplicative adjustment method (i.e. using a retention factor).
- 02.12: Return both results of 02.09 and 02.10 if both adjustment methods are selected.
- 02.13: Upon selection of the adjustment option “all”, return both results of 02.09 and 02.10 as well as the results of an additive and a multiplicative adjustment method that not only adjust the treatment effect but also the threshold value for the decision rule.
- 02.14: Return the same results as in the basic setting if the adjustment parameters are set to 0.5 for the additive method or set to one for the multiplicative method.

As before, in addition to the main results of optimal sample size, optimal threshold value and expected utility, the program should be able to return the following additional data concerning the drug development program:

- 02.15: Return the pre-specified constraint on the total costs as well as the actually calculated costs of phase II and III for the optimal sample size and threshold.
- 02.16: Return the maximum total sample size as well as the separate sample sizes in phase II and III.
- 02.17: Return the probability that the program proceeds to phase III.
- 02.18: Return the probability of a successful program (for every effect size and in total).
- 02.19: Return the number of events in phase II and III (in the time-to-event-setting).
- 02.20: Return the adjustment method and the calculated adjustment parameter.

03. Multitrial – Programs with several phase III trials

The program should also implement a framework developed for phase II/III drug development programs where several phase III trials are performed. This is of particular relevance as regulatory agencies often require statistical significance in two or more phase III trials. Different cases, defined by the number of significant trials needed for approval, should be implemented in the package. For each case, different strategies, defined by the number of phase III trials to be conducted in order to reach the goal of the case, should be implemented. For the success of the drug development program, it is necessary that the treatment effects of all phase III trials point in the same direction. For example, if we select case 3 and strategy 4, we require four phase III trials, where three need to be significant at level α and the treatment effect of the fourth must point in the same direction. Hence, in addition to the parameters from the basic setting, the user should be allowed to provide the following parameters:

- A parameter to set the strategy,
- A parameter to set the case. The following cases and possible strategies should be implemented in the program.

Case	Possible strategies for this case
1	1, 2
2	1 (with significance level of α^2)*, 2, 3, 23 (=2+1)**
3	1 (with significance level of α^3)*, 3, 4

*For cases 2 and 3, the package should also provide the strategy to only use one trial, but with adjusted significance level.

** For case 2, the package should also provide a 2+1 one strategy: If after conducting two trials, only one delivers a significant result and the other trial's treatment effect points at least in the same direction, a third trial should be conducted. This strategy will be called "23" in the package.

In analogy to the other sections, we pose the following requirements:

- 03.01: Calculate the results in the multitrial setting for normally distributed outcome variables.
- 03.02: Calculate the results in the multitrial setting for binary outcome variables.
- 03.03: Calculate the results in the multitrial setting for time-to-event outcome variables.
- 03.04: Upon user selection, calculate the results in the multitrial setting for fixed treatment effects.
- 03.05: Upon user selection, calculate the results in the multitrial setting for treatment effects modeled on a user-specified prior distribution.
- 03.06: Automatically set the internal utility function to -9999 for all sample sizes which exceed the user-defined maximum number of participants in phase II and III. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 03.07: Automatically set the internal utility function to -9999 for all sample sizes whose costs exceed the user-defined maximum cost limit. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 03.08: Automatically set the internal utility function to -9999 for all sample sizes whose drug development program success probability is lower than the user-defined miminum success probability. This should

then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 03.09: Upon user selection, the program should calculate the results using parallel computing, i.e. using more than one core.

For every case and possible strategy, we expect the program to calculate the optimal sample size, the optimal threshold value and the corresponding expected utility of the drug development program correctly. Methods should be implemented according to the table above. If the user sets strategy = TRUE, the program should return every strategy implemented for the specified case. (This should be the default value.) We expect an error message if a strategy is impossible for the respective case. We formulate this as follows:

- 03.10: Calculate the results for the user-selected number of cases using the user-selected strategy.
- 03.11: Upon user request, calculate every implemented strategy for a specific case.
- 03.12: Return an error message if the chosen case and strategy do not match.
- 03.13: Return the same results as in the basic setting if the input parameters match the basic setting, i.e. case = 1 and strategy = 1.
- 03.14: Calculate the results for the 2+1 strategy (i.e. strategy 23 in the packages nomenclature).
- 03.15: Calculate the results for cases 2 and 3 with strategy 1 (using one trial with an adjusted significance level). These results should match the results of the basic setting with manually adjusted significance level to α^2 or α^3, respectively.

As before, in addition to the main results of optimal sample size, the optimal threshold value and expected utility, the program should be able to return the following additional data concerning the drug development program:

- 03.16: Return the pre-specified constraint on the total costs as well as the actually calculated costs of phase II and III for the optimal sample size and threshold.
- 03.17: Return the maximum total sample size as well as the separate sample sizes in phase II and III.
- 03.18: Return the probability that the program proceeds to phase III.
- 03.19: Return the probability of a successful program (for every effect size and in total).
- 03.20: Return the number of events in phase II and III (in the time-to-event-setting).
- 03.21: Return the selected strategy and case.

04. Multiarm – Programs with more than one experimental arm

A further extension implemented are multi-arm trials. So far, only three-arm trials with two experimental treatments and one control are implemented: Assume that two new treatments, e.g. two doses of the same drug, are tested at the same time and that the same control group is included in phase II and phase III. The aim of the phase II/III drug development program is to demonstrate efficacy for at least one of the experimental treatments. In this setting, the user should be able to supply the following parameters in addition to the parameters from the basic setting:

- A parameter to specify which strategy is used, i.e. if only the best candidate or all promising candidates proceed to phase III,
- The event rate of the control arm (in the time-to-event setting).

The best candidate is the treatment group with the highest treatment effect, the promising candidates are all candidates with a treatment effect greater than a pre-specified threshold κ.

As above, possible cost or size constraints should be considered. However in contrast to the above settings, we only implemented fixed treatments effects. Treatment effects modeled on a prior distribution were yet not implemented, but the validation is easily adaptable if this feature is added in the future. Therefore, we expect from the program:

- 04.01: Calculate the results in the multi-arm setting for normally distributed outcome variables.
- 04.02: Calculate the results in the multi-arm setting for binary outcome variables.
- 04.03: Calculate the results in the multi-arm setting for time-to-event outcome variables.

- 04.04: Automatically set the internal utility function to -9999 for all sample sizes which exceed the user-defined maximum number of participants in phase II and III. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 04.05: Automatically set the internal utility function to -9999 for all sample sizes whose costs exceed the user-defined maximum cost limit. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 04.06: Automatically set the internal utility function to -9999 for all sample sizes whose drug development program success probability is lower than the user-defined miminum success probability. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 04.07: Upon user selection, the program should calculate the results using parallel computing, i.e. using more than one core.

The program should be able to calculate the optimal sample size, the optimal threshold value and the expected utility for the possible strategies, i.e. only for the best candidate or all promising candidates (or both). Formulated as requirements this means:

- 04.08: Upon user selection, calculate the results for the strategy where only the best candidates proceed to phase III.
- 04.09: Upon user selection, calculate the results for the strategy where all promising candidates proceed to phase III. As before, in addition to the main results of optimal sample size, the optimal threshold value and the expected utility, the program should be able to return the following additional data concerning the drug development program:
- 04.10: Return the pre-specified constraint on the total costs as well as the actually calculated costs of phase II and III for the optimal sample size and threshold.
- 04.11: Return the maximum total sample size as well as the separate sample sizes in phase II and III.
- 04.12: Return the probability that the program proceeds to phase III.
- 04.13: Return the probability of a successful program (for every effect size and in total).
- 04.14: Return the selected strategy.

05. Multiple endpoints

The program should also provide methods for drug development programs with multiple endpoints. For now, this means that the program provides methods for two endpoints. Moreover, only normally distributed and time-to-event endpoints are implemented in the multiple endpoint setting. (Further extensions may be implemented in the future.) The definition of treatment success is different for the two endpoints:

- In the time-to-event setting, the drug development program is defined to be successful if it proceeds from phase II to phase III and at least one endpoint shows a statistically significant treatment effect in phase III. For example, this situation is found in oncology trials, where overall survival (OS) and progression-free survival (PFS) are the two endpoints of interest.
- For normally distributed endpoints, the drug development program is defined to be successful if it proceeds from phase II to phase III and all endpoints show a statistically significant treatment effect in phase III. For example, this situation is found in Alzheimer's disease trials, where a drug should show significant results in improving cognition (cognitive endpoint) as well as in improving activities of daily living (functional endpoint).

The user should be able to provide the following input values in addition to the general parameters defined in the basic setting:

- The correlation between the two endpoints,
- The event rate of the control arm (in the time-to-event setting), and
- The variances of the endpoints (in the normally distributed setting).

The program should correctly calculate the optimal sample size, the optimal threshold value and the corresponding expected utility for utility-based optimization of phase II/III programs with two time-to-event endpoints. We require the following:

- 05.01: Calculate the results in the multiple endpoints setting for time-to-event outcome variables.
- 05.02: Calculate the results in the multiple endpoints setting for normally distributed outcome variables.
- 05.03: Upon user selection, calculate the results for fixed effects.
- 05.04: Upon user selection, calculate the results for treatment effects modeled on a user-specified prior distribution.
- 05.05: Automatically set the internal utility function to -9999 for all sample sizes which exceed the user-defined maximum number of participants in phase II and III. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 05.06: Automatically set the internal utility function to -9999 for all sample sizes whose costs exceed the user-defined maximum cost limit. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 05.07: Automatically set the internal utility function to -9999 for all sample sizes whose drug development program success probability is lower than the user-defined miminum success probability. This should then lead to different results for the optimal sample size that satisfy the constraints or to a result with a utility of -9999 in case the constraints cannot be satisfied.
- 05.08: Upon user selection, the program should calculate the results using parallel computing, i.e. using more than one core.

As before, in addition to the main results of optimal sample size, optimal threshold value and expected utility, the program should be able to return the following additional data concerning the drug development program:

- 05.09: Return the pre-specified constraint on the total costs as well as the actually calculated costs of phase II and III for the optimal sample size and threshold.
- 05.10: Return the maximum total sample size as well as the separate sample sizes in phase II and III.
- 05.11: Return the probability that the program proceeds to phase III.
- 05.12: Return the probability of a successful program (for every effect size and in total).

The effect size categories small, medium and large can be applied to both endpoints. In order to define an overall effect size from the two individual effect sizes, the package should implement combination rules. For normally distributed endpoints, two different combination rules should be implemented:

- A strict rule assigning a large overall effect in case both endpoints show an effect of large size, a small overall effect in case that at least one of the endpoints shows a small effect, and a medium overall effect otherwise.
- A relaxed rule assigning a large overall effect if at least one of the endpoints shows a large effect, a small effect if both endpoints show a small effect, and a medium overall effect otherwise.

Based on this we require the following features:

- 05.13: Upon user selection, calculate the results for above-explained strict rule regarding treatment effects. This feature is only required for normally distributed endpoints.
- 05.14: Upon user selection, calculate the results for above-explained relaxed rule regarding treatment effects. This feature is only required for normally distributed endpoints.

On the other hand, for time-to-event endpoints, the effect size of the endpoint with larger treatment effect should be selected as overall effect size. In addition, the user should be asked to provide two triples of benefits per category. If only the less important endpoint is significant after phase III, then the smaller benefit triple should be chosen by the software. If at least the more important endpoint is significant, then the larger benefit triple should be chosen by the software. (This combination rule reflects the situation in oncological trials: If only progression-free survival is significant, then a smaller benefit can be expected compared to trials were over all survival is significant.) Based on this we require the following features:

- 05.15: Request two benefit triples from the user. Calculate the results using above-explained combination rule regarding treatment effects and the two user-specified different benefit triples. This feature is only required for time-to-event endpoints.

Test Cases

01. Basic setting test cases

01.01 (shows that req. 01.03, 01.06, 01.12 and 01.15 are met):

Use the function `optimal_tte()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.69 and 0.88,
- event rates of 0.7 for both phase II and phase III,
- the optimization region {10, 11, ..., 400} for the number of participants (events in the time-to-event setting) in phase II,
- the optimization region {0.71, 0.72, ..., 0.95} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 300,000,000$, and 500,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- "fixed=FALSE", i.e. set the function to use a prior distribution,
- weight of 0.3 for the prior distribution,
- amount of information for prior true treatment effect given by 210 and 420 expected events for both treatment effects.

Verify that the function calculates an optimal sample size of 206 in phase II and 354 in phase III (i.e. a total of 560 participants), an expected utility of 432 (in 10^5$), and an optimal threshold value of 0.84 as suggested by Stella Erdmann [2]. Furthermore, verify that one can expect 144 events in phase II and 248 events in phase III (i.e. 392 in total).

01.02 (shows that req. 01.03., 01.05., 01.14 and 01.16 are met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following: Set the parameter "fixed" to be TRUE, thus using fixed assumed treatment effects and set the assumed true treatment effect, i.e. the hazard ratio to 0.8. As we assume fixed true treatment effects this corresponds to setting hr1 = 0.8, hr2 can be set to 0. Set the weight for the prior distribution to be NULL and the number of events to be NULL and NULL. Verify that the function calculates an optimal sample size of 240 in phase II, an expected utility of 352 (in 10^5$) and an optimal threshold value of 0.88 as suggested by Stella Erdmann [2]. Furthermore, verify that the probability to go to phase III is given by 0.73 and the expected number of events in phase II and III is 168 and 546, respectively (714 in total).

01.03 (shows that req. 01.10 and 01.12 are met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following changes and additions: Set the weight for the prior distribution to be 0.6 and set a cost constraint of K=750 (in 10^5$). Verify that the function calculates an optimal sample size of 228 in phase II, an expected utility of 996 (in 10^5$) and an optimal threshold value of 0.84 as suggested by Stella Erdmann [2]. Furthermore, verify that the cost constraint is returned and that the total costs in phase II and III are 271 (in 10^5$) and 478 (in 10^5$).

01.04 (shows that req. 01.09 and 01.13 are met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following changes and additions: Set the weight for the prior distribution to be 0.6 and set a sample size constraint of N=500. Verify that the function calculates an optimal sample size of 170 in phase II and 328 in phase III (i.e. a total sample size of 498), an expected utility of 956 (in 10^5$) and an optimal threshold value of 0.83 as suggested by Stella Erdmann [2].

01.05 (shows that req. 01.11, 01.14 and 01.15 met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following changes and additions: Set the weight for the prior distribution to be 0.6 and set a constraint on the minimal probability of a successful program of S=0.6. Verify that the function calculates an optimal sample size of 470 in phase II, an expected utility of 899 (in 10^5$) and an optimal threshold value of 0.89 as suggested by Stella Erdmann [2]. Furthermore, verify, that the probability to go phase III is given by 0.77 and the probability of a successful program is given by 0.6.

01.06 (shows that req. 01.04 is met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following changes and additions: Set the weight for the prior distribution to be 0.6 and use the option to skip phase II. Verify that the function calculates an optimal sample size of 824 in phase III (corresponding to the total sample size), an expected utility of 1706 (in 10^5$) and an effect size of 0.76 used for planning phase III (returned as `median_prior_HR`, as the classical "optimal" threshold value `HRgo` is returned as `Inf`) as suggested by Stella Erdmann [2].

01.07 (shows that req. 01.07 is met):

Use the function `optimal_tte()`. Supply the same input values as in test case 01.01 to the function except for the following changes and additions: Set the weight for the prior distribution to be 0.6 and use the option to model different population structures and set the parameter γ to 0.025. Verify that the function calculates an optimal sample size of 310 in phase II, an expected utility of 1207 (in 10^5$) and an optimal threshold value of 0.86 as suggested by Stella Erdmann [2].

01.08 (shows that req. 01.02, 01.05 and 01.12 are met):

Use the function `optimal_binary()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment rate of 0.6 in the control group and assumed true rates of 0.5 and NULL for the prior distribution of the treatment group,
- the optimization region of all even numbers {10, 12, ..., 500} for the number of participants in phase II,
- the optimization region {0.7, 0.71, ..., 0.9} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 300,000,000$, and 500,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- `fixed=TRUE`, i.e. set the function to use fixed treatment effects not modelled on a prior distribution,
- weight of NULL for the prior distribution,
- NULL and NULL events for both treatment effects.

Verify that the function calculates an optimal sample size of 204, an expected utility of 299 (in 10^5$) and an optimal threshold value of 0.90 as suggested by Stella Erdmann [2]. Furthermore, verify that the programs returns the cost constraint (Inf in this case) as well as the total costs in phase II and III, 253 (in 10^5$) and 769 (in 10^5$), respectively.

01.09 (shows that req. 01.02 and 01.06 are met):

Use the function `optimal_binary()`. Supply the same input values as in test case 01.08 to the function except for the following changes and additions: Set the parameter "fixed" to be FALSE. Set the weight for the prior distribution to be 0.4, the assumed true rates for the prior distribution of the treatment group to be 0.3 and 0.5 and the number of events to be 30 and 60. Verify that the function calculates an optimal sample size of 224, an expected utility of 1542 (in 10^5$) and an optimal threshold value of 0.89 as suggested by Stella Erdmann [2].

01.10 (shows that req. 01.01 and 01.06 are met):

Use the function `optimal_normal()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.375 and 0.5,
- the optimization region of even numbers $\{10, 12, \ldots, 500\}$ for the number of participants in phase II,
- the optimization region $\{0.01, 0.02, \ldots, 0.5\}$ for the threshold values,
- boundaries of 0, 0.375 and 0.625 for the effect size categories small, medium and large,
- expected gains of 62,500,000$, 200,000,000$ and 1,000,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 1,500,000$ in phase II and of 2,000,000$ in phase III,
- variable costs of 67,500$ in phase II and 72,000$ in phase III,
- "fixed=FALSE", i.e. set the function to model the treatment effects on a prior distribution,
- weight of 0.5 for the prior distribution,
- 300 and 600 expected events for both treatment effects,
- truncation values of a = 0 and b = 0.75.

Verify that the function calculates an optimal sample size of 86 in phase II, an expected utility of 337 (in 10^5$) and an optimal threshold value of 0.19 as suggested by Stella Erdmann [2].

01.11 (shows that req. 01.01, 01.05 and 01.15 are met):

Use the function `optimal_normal()`. Supply the same input values as in test case 01.10 to the function except for the following changes and additions: Set the parameter "fixed" to be TRUE and set the value for the assumed true treatment effect to 0.625 and NULL. Set the weight for the prior distribution to be NULL, the number of events to be NULL and NULL and the truncation values to be a = NULL and b = NULL. Verify that the function calculates an optimal sample size of 78 in phase II, an expected utility of 944 (in 10^5$) and an optimal threshold value of 0.12 as suggested by Stella Erdmann [2]. Furthermore, verify that the probability of a successful program is given by 0.83, which is the sum of the probabilities of a small (0.51), medium (0.30) or large (0.02) treatment effect.

01.12 (shows that req. 01.08 is met):

Use the `function optimal_tte`. Supply the same input values as in test case 01.01 to the function except for the following change: Set the number of cores for parallel computing to 6. Verify that the computation time will increase compared to the setting in 01.01.

01.13 (shows that req. 01.01, 01.13, 01.14 are met):

Use the function `Epgo_normal()`. Supply 10 sets of the following input values:

- threshold values κ as parameter `kappa`,
- sample sizes of phase II n_2 as parameter `n2`,
- assumed true treatment effects Δ as parameter `Delta1`,
- `fixed=TRUE`, i.e. set the function to use fixed treatment effects not modelled on a prior distribution, and
- `NULL` for all other input values of the function.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the probability formula

$$p_{go}^{\Delta} = \mathrm{P}(\hat{\Delta}_2 \geq \kappa|\Delta) = \Phi\left(\frac{\Delta - \kappa}{\sqrt{4/n_2}}\right)$$

from (Preussler 2020), where Φ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$.

Use the function `En3_normal()`. Supply 10 sets of the following input values:

- the same input values as above,
- significance levels α as parameter `alpha`, and
- type-II error levels β as parameter `beta`.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the sample size formula

$$\mathrm{E}[N_3^{\Delta}(\hat{\Delta}_2) \cdot 1_{\{\hat{\Delta}_2 \geq \kappa\}}] = \int_{\kappa}^{\infty} N_3^{\Delta}(\hat{\Delta}_2) \cdot f(\hat{\Delta}_2) d\hat{\Delta}_2,$$

where

$$N_3^{\Delta,\tau} = N_3^{\Delta,\tau}(\hat{\Delta}_2) = \frac{4 \cdot (z_{1-\alpha} + z_{1-\beta})^2}{(\hat{\Delta}_2 - \tau/\sigma)^2},$$

where $\tau/\sigma = 0$ (i.e. testing for superiority) and $f(\hat{\Delta}_2)$ is the probability density function of $\mathcal{N}(\mu = \Delta, \sigma^2 = 4/n_2)$. This formula is the fixed case (no prior distribution) of eq. 2.8 from (Preussler 2020). (Note: R and SAS use the standard deviation for specifying normal distribution. Here, we specify using the variance.)

01.14 (shows that req. 01.02, 01.13, 01.14 are met):

Use the function `Epgo_binary()`. Supply 10 sets of the following input values:

- threshold risk ratios RR_{go} as parameter `RRgo`,
- sample sizes of phase II n_2 as parameter `n2`,
- assumed true rate in the control group p_0 as parameter `p0`,
- assumed true rate in the treatment group p_1 as parameter `p11`,
- `fixed=TRUE`, i.e. set the function to use fixed treatment effects not modelled on a prior distribution, and
- `NULL` for all other input values of the function.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the probability formula

$$p_{go}^{\varrho} = \Phi\left(\frac{\varrho - \kappa}{\sqrt{2/n_2 \cdot (\frac{1-p_0}{p_0} + \frac{1-p_1}{p_1})}}\right)$$

from (Preussler 2020) where $\varrho = -\log(p_{11}/p_0)$ and $\kappa = -\log(RR_{go})$ and Φ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$.

Use the function `En3_binary()`. Supply 10 sets of the following input values:

- the same input values as above,
- significance levels α as parameter `alpha`, and
- type-II error levels β as parameter `beta`.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the sample size formula

$$\mathrm{E}[N_3^{\varrho}(\hat{\varrho}_2) \cdot 1_{\{\hat{\varrho}_2 \geq \kappa\}}] = \int_{\kappa}^{\infty} N_3^{\varrho}(\hat{\varrho}_2) \cdot f(\hat{\varrho}_2) d\hat{\varrho}_2,$$

where

$$N_3^{\varrho}(\hat{\varrho}_2) = \frac{2 \cdot \left(z_{1-\alpha} \cdot \sqrt{\frac{2 \cdot (1-p)}{p}} + z_{1-\beta} \cdot \sqrt{\frac{1-p_0}{p_0} + \frac{1-p_1}{p_1}} \right)^2}{\hat{\varrho}_2^2},$$

and $f(\hat{\varrho}_2)$ is the probability density function of the normal distribution $\mathcal{N}(\mu = \varrho, \sigma^2 = 2/n_2 \cdot (\frac{1-p_0}{p_0} + \frac{1-p_1}{p_1}))$, which is the fixed case (no prior distribution) of eq. 2.8 from (Preussler 2020).

01.15 (shows that req. 01.03, 01.13, 01.14 are met):

Use the function `Epgo_tte()`. Supply 10 sets of the following input values:

- threshold values HR_{go} as parameter `HRgo`,
- number of events in phase II d_2 as parameter `d2`,
- assumed true treatment effects HR as parameter `hr1`,
- `fixed=TRUE`, i.e. set the function to use fixed treatment effects not modelled on a prior distribution, and
- `NULL` for all other input values of the function.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the probability formula

$$p_{go}^{\theta} = \mathrm{P}(\hat{\theta}_2 \geq \kappa | \theta) = \Phi \left(\frac{\theta - \kappa}{\sqrt{4/d_2}} \right),$$

from (Preussler 2020), where $\theta = -\log(HR)$, $\kappa = -\log(HR_{go})$ and Φ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$.

Use the function `Ed3_tte()`. Supply 10 sets of the following input values:

- the same input values as above,
- significance levels α as parameter `alpha`, and
- type-II error levels β as parameter `beta`.

Calculate the function output for all 10 parameter sets and compare the results to the results of a SAS program implementing the event number formula

$$d_3 = \int_{\kappa}^{\infty} D_3(\hat{\theta}_2) \cdot f(\hat{\theta}_2) \; d\hat{\theta}_2$$

with

$$D_3 = \frac{4 \cdot (z_{1-\alpha} + z_{1-\beta})^2}{\hat{\theta}_2^2}$$

and $f(\hat{\theta}_2)$ being probability density function of the normal distribution $\mathcal{N}(\mu = \theta, \sigma^2 = 4/d_2)$, which is the fixed case (no prior distribution) of eq. 2.7 from (Preussler 2020).

02. Bias adjustment test cases

02.01 (shows that req. 02.03, 02.05, 02.10 and 02.19 are met):

Use the function `optimal_bias()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.69 and 0.88,
- event rates of 0.7 for both phase II and phase III,
- the optimization region {20, 25, ..., 100} for the number of events in phase II,
- the optimization region {0.7, 0.72, ..., 0.9} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,

- expected gains of 100,000,000$, 200,000,000$, and 300,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- "fixed=FALSE", i.e. set the function to use a prior distribution,
- weight of 0.3 for the prior distribution,
- amount of information for prior true treatment effect given by 210 and 420 events for each treatment effect in phase II,
- choice of "additive" bias adjustment method,
- the optimization region $\{0.3, 0.325, \ldots, 0.5\}$ for the additive adjustment parameter α_{CI},
- value of NULL for the multiplicative adjustment parameter.

Verify that the function calculates an optimal sample size of 122 in phase II and 200 in phase III (i.e. a total of 322 participants), an expected utility of 78 (in 10^5$), and an optimal threshold value of 0.78 as suggested by Stella Erdmann [2]. Furthermore, verify that one can expect 85 events in phase II and 140 events in phase III (i.e. 225 in total).

02.02 (shows that req. 02.03, 02.05, 02.11 and 02.17 are met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.01, however set the adjustment method to "mulitplicative" and set the optimization region for the multiplicative adjustment parameter λ to $\{0.5, 0.55, \ldots, 1\}$ and the parameters for the additive method to `NULL`.

Verify that the function calculates an optimal sample size of 136 in phase II and 244 in phase III, i.e. a total of 380 participants), an expected utility of 99 (in 10^5$), and an optimal threshold value of 0.76 as suggested by Stella Erdmann [2]. Furthermore, verify that the probability to go to phase III is 0.38.

02.03 (shows that req. 02.06, 02.12, 02.16 and 02.20 are met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.01, however set the adjustment method to "both" and set the optimization region for the multiplicative adjustment parameter λ to $\{0.5, 0.55, \ldots, 1\}$ and the parameters for the additive method α_{CI} to $\{0.3, 0.325, \ldots, 0.5\}$. Furthermore, set a constraint for the maximum sample size to be 350.

Verify that the program returns the results for both adjustment methods by returning the selected methods "multipl." and "add." as well as the calculated adjustment parameter. Hereby verify, that the results for the additive method are the same as in test case 02.01 as the sample size constraint is not binding and that the optimal sample size for the multiplicative method changes to 100 in phase II and 240 in phase III, (i.e. a total of 340) and the expected utility changes to 98 (in 10^5$).

02.04 (shows that req. 02.13 and 02.20 are met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.03 (including the optimization regions for the adjustment parameters and the constraint for the maximal sample size), however set the adjustment method to "all".

Verify that the program returns the results for both adjustment methods by returning the selected method as well as the calculated adjustment parameter and further returns the results of an additive and a multiplicative adjustment method that not only adjust the treatment effect but also the threshold value for the decision rule. Hereby verify that the results for the basic additive and multiplicative method the same as in test case 02.03. Moreover, verify that for the advanced method, the program returns an expected overall utility of 96.69 (in 10^5$), an adjustment parameter of 0.75 and optimal sample sizes of 108 in phase II and 200 in phase III (i.e. an overall sample size of 326) for the advanced multiplicative method ("multipl2") and an expected overall utility of 77.40 (in 10^5$), an adjustment parameter of 0.475 and optimal sample sizes of 136 in phase II and 206 in phase III (i.e. an overall sample size of 342) for the advanced additive method ("add2").

02.05 (shows that req. 02.04, 02.07 and 02.15 are met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.01 (including the optimization regions for the additive adjustment parameter), however set the parameter fixed to be TRUE, thus using a fixed treatment effect. Redo this, however the second time set a cost constraint of 40,000,000$.

Verify that the expected utility changes from 865.02 (in 10^5$) to 474.18 (in 10^5$) and the optimal sample size changes from 144 to 44 and from 478 to 172 in phase II and III respectively due to the cost constraint. The optimal adjustment parameter changes from 0.5 to 0.475. Furthermore verify, that the costs in phase II and III are 208 (in 10^5$) and 608 (in 10^5$) without the constraint and 133 (in 10^5$) and 263 (in 10^5$) with the constraint, i.e. the cost constraint is met at the optimal result.

02.06 (shows that req. 02.08 and 02.18 are met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.01 (including the optimization regions for the additive adjustment parameter), however set the parameter fixed to be TRUE, thus using a fixed treatment effect. Redo this, however the second time set a constraint of 0.7 for the minimal success probability.

Verify that the expected utility changes from 865.02 (in 10^5$) to 857.77 (in 10^5$) and the optimal sample size changes from 144 to 144 and from 478 to 552 in phase II and III respectively due to the probability constraint. Verify that the optimal adjustment parameter changes from 0.5 to 0.5. Furthermore verify that without the constraint, the probability of a successful program is 0.68, with a probability of 0.07, 0.21 and 0.39 for small, medium or large treatment effects; and that with the constraint, the probability of a successful program is 0.7, with a probability of 0.07, 0.21 and 0.42 for small, medium or large treatment effects. This means that the cost constraint is met at the optimal result.

02.07 (shows that req. 02.14 is met):

Use the function `optimal_bias`. Supply the same input values as in test case 02.01, however change the setting, such that the optimization region for the additive adjustment parameter just contains the point $\{0.5\}$ and the adjustment parameter for the multiplicative adjustment method just contains the point $\{1\}$. Furthermore, change the adjustment method to `"both"`. Then, use the function `optimal_tte` and supply the same input values as in test case 02.01.

Verify that both adjustment methods and the case with no bias adjustment return the same results, i.e. an expected utility of 75.8 (in 10^5$) and optimal sample sizes of 122 participants in phase II and 210 participants in phase III (i.e. a total sample size of 332).

02.08 (shows that req. 02.09 is met):

Test case 02.09 was omitted for the following reason: A decrease in computation time can only be noted in processes with longer run time. In this setting with a rather short run time, the overhead of distributing tasks to more kernels actually leads to a slight increase in run time. As this feature is not critical and its general functionality is also checked in other test cases, we simple omit the test case. The original text read as follows:

Use the function `optimal_bias`. Supply the same input values as in test case 02.01, however change the number of clusters for parallel computing to 6. Verify that the computation time will increase compared to the setting in 02.01.

02.09 (shows that req. 02.01, 02.05 and 02.11 are met):

Use the function `optimal_bias_normal()`. Supply the following input values to the function:

- a significance level of 0.05,
- a power of 0.9, i.e. β of 0.1,
- assumed true prior treatment effects of 0.625 and 0.325,

- the optimization region of numbers {20, 24, ..., 400} for the number of participants in phase II,
- the optimization region {0.02, 0.04,..., 0.4} for the threshold values,
- boundaries of 0, 0.5 and 0.8 for the effect size categories small, medium and large,
- expected gains of 300,000,000$, 800,000,000$ and 1,000,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 1,500,000$ in phase II and of 2,000,000$ in phase III,
- variable costs of 67,500$ in phase II and 72,000$ in phase III,
- "fixed=FALSE", i.e. set the function to model the treatment effects on a prior distribution,
- weight of 0.5 for the prior distribution,
- 300 and 600 events as the amount of information for the two prior treatment effect estimates, respectively,
- truncation values of a = 0.25 and b = 0.75,
- multiplicative adjustment method, and
- optimization region of {0.7, 0.71, ..., 0.9} for the adjustment parameter λ .

Verify that the function returns an expected utility of 2899.11 (in 10^5$), an optimal threshold value of 0.12 and an optimal sample size of 192 in phase II and 474 in phase III (i.e. 666 in total).

02.10 (shows that req. 02.01, 02.04 and 02.13 are met):

Use the function `optimal_bias_normal()`. Supply the same input values as in test case 02.09 (including the optimization regions for the multiplicative adjustment parameter), however set the parameter fixed to `"TRUE"`. Furthermore use the adjustment method "all" and provide the following optimization set for the additive adjustment parameter α_{CI}: {0.25, 0.275, ..., 0.5}.

Verify that the program returns the results for both adjustment methods by returning the selected method as well as the calculated adjustment parameter and further returns the results of an additive and a multiplicative adjustment method that not only adjust the treatment effect but also the threshold value for the decision rule. Verify that for the basic multiplicative method, the program returns an expected overall utility of 3861.76 (in 10^5$), an adjustment parameter of 0.7 and optimal sample sizes of 88 in phase II and 310 in phase III (i.e. an overall sample size of 398). Verify that for the basic additive method, it returns an expected overall utility of 3631.51 (in 10^5$), an adjustment parameter of 0.25 and optimal sample sizes of 96 in phase II and 306 in phase III (i.e. an overall sample size of 402). Moreover, verify that for the advanced multiplicative method ("multipl2"), the program returns an expected overall utility of 3860.12 (in 10^5$), an adjustment parameter of 0.7 and optimal sample sizes of 88 in phase II and 306 in phase III (i.e. an overall sample size of 394). Finally, verify that for the advanced additive method ("add2"), the program returns an expected overall utility of 3631.28 (in 10^5$), an adjustment parameter of 0.25 and optimal sample sizes of 96 in phase II and 312 in phase III (i.e. an overall sample size of 408).

02.11 (shows that req. 02.02, 02.05 and 02.10 are met):

Use the function `optimal_bias_binary()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of p0 = 0.6, p11 = 0.3, p12= 0.5,
- the optimization region of all even numbers {10, 12, ..., 500} for the number of participants in phase II,
- the optimization region {0.7, 0.71, ..., 0.9} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000, 200,000,000, and 300,000,000 for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- "fixed=FALSE", i.e. set the function to use treatment effects modeled on a prior distribution,
- weight of 0.3 for the prior distribution,
- a sample size of 30 and 60 for the two treatment effect estimate, respectively,
- additive adjustment method "additive", and

- an optimization region of $\{0.1, 0.125, \ldots, 0.5\}$ for the adjustment parameter α_{CI}.

Verify that the function calculates an optimal sample size of 166 in phase II and 264 in phase III (i.e. a total of 430 participants), an expected utility of 605.91 (in 10^5$), and an optimal threshold value of 0.82 as well as an optimal additive adjustment parameter of 0.275.

02.12 (shows that req. 02.02, 02.04 and 02.12 are met):

Use the function `optimal_bias_binary()`. Supply the same input values as in test case 02.11 (including the optimization region for the additive adjustment parameter), however set the parameter fixed to `"TRUE"`. Furthermore use the adjustment method "both" and provide the following optimization set for the multiplicative adjustment parameter λ: $\{0.5, 0.55, \ldots, 1\}$.

Verify that the program returns the results for both adjustment methods by returning the selected method as well as the calculated adjustment parameter. Hereby verify that for the multiplicative method, the function calculates an optimal sample size of 206 in phase II and 340 in phase III (i.e. a total of 546 participants), an expected utility of 2116.67 (in 10^5$), and an optimal threshold value of 0.8 as well as an optimal multiplicative adjustment parameter of 0.65. Furthermore, verify that the for the additive method, the function calculates an optimal sample size of 190 in phase II and 226 in phase III (i.e. a total of 416 participants), an expected utility of 1996.10 (in 10^5$), and an optimal threshold value of 0.77 as well as an optimal additive adjustment parameter of 0.1.

03. Multitrial test cases

03.01 (shows that req. 03.03, 03.05, 03.11 and 03.20 are met):

Use the function `optimal_multitrial`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.69 and 0.88,
- event rates of 0.7 for both phase II and phase III,
- the optimization region $\{100, 104, \ldots, 300\}$ for the number of events in phase II,
- the optimization region $\{0.65, 0.71, \ldots, 0.8\}$ for the threshold values,
- expected gains of 100,000,000$, 300,000,000$, and 500,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- `fixed=FALSE`, i.e. set the function to use a prior distribution,
- weight of 0.3 for the prior distribution,
- amount of information for prior true treatment effect given by 210 and 420 expected events for the two assumed treatment effects, respectively,
- use case 2 (i.e. at least two trials have to show a significant positive treatment effect), and
- use `strategy = TRUE`, i.e. calculating all implemented strategies for the specified case.

Verify that for strategy 1, the program returns an expected utility of -1.89, optimal sample sizes of 200 in phase II and 238 in phase III (i.e. 438 in total), 140 events in phase II and 167 events in phase III (i.e. 307 in total), and an optimal threshold value of 0.75.

For strategy 2, the program returns an expected utility of -94.31, optimal sample sizes of 172 in phase II and 172 in phase III (i.e. 344 in total), corresponding to two trials with 86 participants each, 120 events in phase II and 122 event in phase III (i.e. 242 in total), and an optimal threshold value of 0.72.

For strategy 3, the program returns an expected utility of -12.88, optimal sample sizes of 224 in phase II and 252 in phase III, (i.e. 476 in total) corresponding to three trials with 84 participants each, 156 events in phase II and 177 event in phase III (i.e. 333 in total), and an optimal threshold value of 0.72.

For strategy 23, the program returns an expected utility of 45.72, optimal sample sizes of 178 in phase II and

194 in phase III (i.e. 372 in total), 124 events in phase II and 136 events in phase III (i.e. 260 in total), and an optimal threshold value of 0.73. Furthermore, the probability, that a third trial is needed is given by 0.09.

03.02 (shows that req. 03.03, 03.10, 03.15 and 03.18 are met):

Use the function `optimal_multitrial`. Supply the same input values as in test case 03.01, however, set the parameter case to 3 and the parameter strategy to 1.

Verify that the program returns an optimal sample size of 160 in phase II and 130 in phase III (i.e. a total number of 290 participants), an expected utility of -148.57 and an optimal threshold value of 0.67. Furthermore, verify, that the probability to go to phase III is 0.2.

03.03 (shows that req. 03.03, 03.07 and 03.16 are met):

Use the function `optimal_multitrial`. Supply the same input values as in test case 03.01, however set a cost constraint of 50,000,000 $.

Verify that for strategy 1 the program returns an expected utility of -4.21, optimal sample sizes of 172 in phase II and 212 in phase III (i.e. 384 in total), costs of 229 (in 10^5 $) in phase II and 261 (in 10^5 $) in phase III, and an optimal threshold value of 0.74. For strategy 2, the results do not change compared to test case 03.01, as the cost constraint is not binding. For strategy 3, the program returns an expected utility of -27.31, optimal sample sizes of 172 in phase II and 156 in phase III, (i.e. 328 in total) corresponding to three trials with 52 participants each, costs of 229 (in 10^5 $) in phase II and 254 (in 10^5 $) in phase III, and an optimal threshold value of 0.68. For strategy 23, the results do not change compared to test case 03.01, as the cost constraint is not binding.

03.04 (shows that req. 03.12 is met):

Use the function `optimal_multitrial`. Supply the same input values as in test case 03.01, however change the parameter case to 3 and the parameter strategy to 2. Verify that the program returns an ERROR.

03.05 (shows that req. 03.04, 03.14, 03.20 and 3.21 are met):

Use the function `optimal_multitrial`. Supply the same input values as in test case 03.01, however set the parameter `fixed` to be "TRUE" and change the optimization region to $\{200, 204, \ldots, 400\}$ for the number of events in phase II and to $\{0.8, 0.81, \ldots, 0.88\}$ for the threshold values

Verify that for strategy 1, the program returns an expected utility of 1165.47, optimal sample sizes of 424 in phase II and 1044 in phase III (i.e. 1468 in total), 296 events in phase II and 731 event in phase III (i.e. 1027 in total), and an optimal threshold value of 0.86.

For strategy 2, the program returns an expected utility of 810.67, optimal sample sizes of 384 in phase II and 1040 in phase III (i.e. 1424 in total), corresponding to two trials with 520 participants each, 268 events in phase II and 728 event in phase III (i.e. 996 in total), and an optimal threshold value of 0.85.

For strategy 3, the program returns an expected utility of 1045.28, optimal sample sizes of 446 in phase II and 1398 in phase III, (i.e. 1844 in total) corresponding to three trials with 466 participants each, 312 events in phase II and 978 event in phase III (i.e. 1290 in total), and an optimal threshold value of 0.82.

For strategy 23, the program returns an expected utility of -47.41, optimal sample sizes of 286 in phase II and 454 in phase III, (i.e. 740 in total), 200 events in phase II and 318 event in phase III (i.e. 518 in total), and an optimal threshold value of 0.80.

03.06 (shows that req. 03.13 and 03.20 are met):

Use the function `optimal_multitrial`. Supply the same input values as in test case 01.01 and set the case and the strategy to 1.

Verify, that the function returns the same results as in test case 01.01. i.e. an optimal sample size of 206 in phase II and 354 in phase III (i.e. a total of 560 participants), an expected utility of 432 (in 10^5$), and an optimal threshold value of 0.84 as suggested by Stella Erdmann [2]. Furthermore, verify that one can expect 144 events in phase II and 248 events in phase III (i.e. 392 in total).

03.07 (shows that req. 03.02, 03.05 and 03.11 are met):

Use the function `optimal_multitrial_binary()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment rate of 0.6 in the control group and assumed true rates of 0.3 and 0.5 for the prior distribution of the treatment group,
- the optimization region of all even numbers {100, 104, ..., 400} for the number of participants in phase II,
- the optimization region {0.7, 0.71, ..., 0.8} for the threshold values,
- expected gains of 100,000,000$, 300,000,000$, and 500,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- `fixed=FALSE`, i.e. set the function to use treatment effects modelled on a prior distribution,
- weight of 0.3 for the prior distribution,
- sample sizes of 30 and 60 as the amount of information for the two assumed treatment effects,
- use case 3 (i.e. at least three trials have to show a significant positive treatment effect), and
- use `strategy = TRUE`, hence calculating all implemented strategies for the specified case.

Verify that for strategy 1, the program returns an expected utility of 405.56 (in 10^5$), an optimal threshold value of 0.77 and an optimal number of participants of 176 in phase II and 306 in phase III (i.e. 482 in total). Note that the value for `alpha` was automatically changed to `alpha^3` by the program.

For strategy 3, the program returns an expected utility of 1494.31 (in 10^5$), an optimal threshold value of 0.77 and an optimal number of participants of 340 in phase II and 336 (corresponds to three trials with 112 participants) in phase III (i.e. 676 in total).

For strategy 4, the program returns an expected utility of 1736.36 (in 10^5$), an optimal threshold value of 0.76 and an optimal number of participants of 384 in phase II and 408 (corresponds to four trials with 102 participants) in phase III (i.e. 792 in total).

03.08 (shows that req. 03.02 and 03.14 are met):

Use the function `optimal_multitrial_binary`. Supply the same input values as in test case 03.07, however change the case to 2 and the strategy to 23, thus, if after conducting two trials, only one delivers a significant result and the other trial's treatment effect points at least in the same direction, a third trial is conducted.

Verify that the program returns an expected utility of 1701.93 (in 10^5$) and an optimal number of participants of 262 in phase II and 224 in phase III.

03.09 (shows that req. 03.02, 03.04 and 03.21 are met):

Use the function `optimal_multitrial_binary`. Supply the same input values as in test case 03.07, however change the parameter fixed to be `"TRUE"` and the optimization region for the threshold values to {0.8, 0.81, ..., 0.95} and set the case to 1 and the strategy to `"TRUE"`, hence calculating all implemented strategies for the specified case.

Verify that for strategy 1, the program returns an expected utility of 3187.57 (in 10^5$), an optimal threshold value of 0.91 and an optimal number of participants of 176 in phase II and 158 in phase III (i.e. 334 in total).

For strategy 2, the program returns an expected utility of 3657.70 (in 10^5$), an optimal threshold value of 0.89 and an optimal number of participants of 240 in phase II and 284 (corresponds to two trials with 142 participants) in phase III (i.e. 524 in total).

03.10 (shows that req. 03.06, 03.10 and 03.17 are met):

Use the function `optimal_multitrial_binary`. Supply the same input values as in test case 03.07, however change the parameter fixed to be `"TRUE"` and the optimization region for the threshold values to {0.8, 0.81, ..., 0.95} and set the parameter case to 2 and the strategy to 3. Redo this, however, the second time set a sample size constraint of 600.

Verify that the expected utility changes from 2923.29 to 1795.38 (in 10^5$), and the optimal sample size changes from 288 to 132 in phase II and from 408 to 468 in phase III (in total it changes from 656 to 600).

03.11 (shows that req. 03.01, 03.05, 03.11 and 03.15 are met):

Use the function `optimal_multitrial_normal()`. Supply the following input values to the function:

- a significance level of 0.05,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.375 and 0.5,
- the optimization region {200, 204, ..., 500} for the number of participants in phase II,
- the optimization region {0.1, 0.12,..., 0.2} for the threshold values,
- expected gains of 300,000,000$, 800,000,000$ and 1,000,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 1,500,000$ in phase II and of 2,000,000$ in phase III,
- variable costs of 67,500$ in phase II and 72,000$ in phase III,
- `fixed=FALSE`, i.e. set the function to model the treatment effects on a prior distribution,
- weight of 0.5 for the prior distribution,
- sample sizes of 300 and 600 as the amount of information on which the two assumed treatment effects are based,
- truncation values of a = 0.25 and b = 0.75,
- use case 3 (i.e. at least three trials have to show a significant positive treatment effect), and
- use `strategy = TRUE`, i.e. calculate all implemented strategies for the specified case.

Verify that for strategy 1, the program returns an expected utility of 1660.95 (in 10^5$), an optimal threshold value of 0.16 and an optimal number of participants of 388 in phase II and 666 in phase III (i.e. 1054 in total). Moreover, the program returns a probability of a successful program of 0.81.

For strategy 3, the program returns an expected utility of 1282.14 (in 10^5$), an optimal threshold value of 0.16 and an optimal number of participants of 332 in phase II and 702 in phase III (i.e. 1034 in total), corresponding to three trials with 234 participants each. Moreover, the program returns a probability of a successful program of 0.68.

For strategy 4, the program returns an expected utility of 1786.02 (in 10^5$), an optimal threshold value of 0.18 and an optimal number of participants of 356 in phase II and 888 in phase III (i.e. 1244 in total), corresponding to four trials with 222 participants each. Moreover, the program returns a probability of a successful program of 0.85.

03.12 (shows that req. 03.09 is met):

Use the function `optimal_multitrial_normal()`. Supply the same input values as in test case 03.11, however change the number of clusters used for parallel computing from 12 to 6. Verify that the computation time will increase compared to the setting in 03.11.

03.13 (shows that req. 03.01, 03.08 and 03.19 are met):

Use the function `optimal_multitrial_normal()`. Supply the same input values as in test case 03.11, however change the parameter fixed to be `"TRUE"`. Redo this, however, the second time set a minimum success probability of 0.82.

Verify that for strategy 1 without the constraint, the program returns an expected utility of 1514.35 (in 10^5$), an optimal threshold value of 0.16 and an optimal number of participants of 440 in phase II and 830 in phase III (i.e. 1270 in total). Moreover, the program returns a probability of a successful program of 0.81.

For strategy 3 without the constraint, the program returns an expected utility of 1116.60 (in 10^5$), an optimal threshold value of 0.16 and an optimal number of participants of 364 in phase II and 882 in phase III (i.e. 1246 in total), corresponding to three trials with 294 participants each. Moreover, the program returns a probability of a successful program of 0.68.

For strategy 4 without the constraint, the program returns an expected utility of 1395.35 (in 10^5$), an optimal threshold value of 0.18 and an optimal number of participants of 424 in phase II and 1128 in phase III (i.e. 1270 in total), corresponding to four trials with 282 participants each. Moreover, the program returns a probability of a successful program of 0.86.

For strategy 1 under the constraint, the expected utility changes to 1513.26 (in 10^5$), the optimal threshold value to 0.14 and an optimal number of participants of 444 in phase II and 852 in phase III (i.e. 1296 in total). Moreover, the program now returns a probability of a successful program of 0.82.

For strategy 3 under the constraint, the constraint can not be met withing the optimization region, so the program returns an expected utility of -9999.

For strategy 4 under the constraint, the program returns the same results as before as the constraint is fulfilled at the optimum for this strategy.

03.14 (shows that req. 03.01, 03.04 and 03.10 are met):

Use the function `optimal_multitrial_normal()`. Supply the same input values as in test case 03.11, however change the parameter fixed to be `"TRUE"` and use case 2 (i.e. at least three trials have to show a significant positive treatment effect) and the strategy 3 (i.e. three trials are conducted in phase III).

Verify that the program returns an expected utility of 1749.97 (in 10^5$), an optimal threshold value of 0.16 and an optimal number of participants of 416 in phase II and 876 in phase III (i.e. 1292 in total), corresponding to three trials with 292 participants each.

04. Multiarm test cases

04.01 (shows that req. 04.03, 04.08 and 04.12 are met):

Use the function `optimal_multiarm`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment hazard ratios of 0.75 and 0.85,
- event rate of 0.6 in the control arm,
- the optimization region $\{10, 11, \ldots, 200\}$ for the number of events in phase II,
- the optimization region $\{0.71, 0.72, \ldots, 0.9\}$ for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 200,000,000$, and 300,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- use strategy 1, i.e. the strategy where only the best of the promising treatments proceeds to phase III.

Verify that the program returns an optimal utility of 8.56 (in 10^5$), an optimal sample size of 141 in phase II and 391 in phase III (i.e. a total of 532) and an optimal threshold value for the go-decision of 0.82 as suggested by Stella Erdmann [2]. Furthermore, verify that the probability to go to phase III is given by 0.72.

04.02 (shows that req. 04.03, 04.09 and 04.12 are met):

Use the function `optimal_multiarm`. Supply the same input values as in test case 04.01, however use strategy 2, i.e. the strategy where all promising treatments proceed to phase III.

Verify that the program returns an optimal utility of 4.36 (in 10^5$), an optimal sample size of 79 in phase II and 426 in phase III (i.e. a total of 505) and an optimal threshold value for the go-decision of 0.78 as suggested by Stella Erdmann [2]. Furthermore, verify that the probability to go to phase III is given by 0.65.

04.03 (shows that req. 04.03, 04.04 and 04.14 are met):

Use the function `optimal_multiarm`. Supply the same input values as in test case 04.01, however set the parameter strategy to 3, i.e. calculate the results for both strategies. Furthermore, set a sample size constraint of 520.

Verify that the program returns the results for both strategies, the results for strategy 2 are the same as in test case 04.02. as the constraint is not binding, however, the results strategy 1 change as follows: The optimal utility changes to 8.34, the optimal sample size changes to 133 in phase II and 383 in phase III (i.e. a total of 516) and the optimal threshold value for the go-decision is 0.82.

04.04 (shows that req. 04.07 is met):

Use the function `optimal_multiarm`. Supply the same input values as in test case 04.02, however change the number of cores for parallel computing to 6.

Verify that the computation time will increase compared to the setting in 04.02.

04.05 (shows that req. 04.02, 04.08 and 04.14 are met):

Use the function `optimal_multiarm_binary()`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment rate of 0.5 in the control group and assumed true rates of 0.3 and 0.4 for the two treatment arms,
- the optimization region of all even numbers {200, 202, ..., 400} for the number of participants in phase II,
- the optimization region {0.7, 0.71, ..., 0.9} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 200,000,000$, and 300,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III, and
- strategy 1, i.e. the strategy where only the best promising treatment proceeds to phase III.

Verify that the program returns an optimal utility of 1264.64 (in 10^5$), an optimal sample size of 386 in phase II and 344 in phase III (i.e. a total of 730) and an optimal threshold value for the go-decision of 0.86.

04.06 (shows that req. 04.02, 04.09 and 04.14 are met):

Use the function `optimal_multiarm_binary`. Supply the same input values as in test case 04.05, however set the parameter strategy to 2, i.e. calculating the results if all promising treatments proceed to phase III.

Verify that the program returns an optimal utility of 1281.74 (in 10^5$), an optimal sample size of 312 in phase II and 561 in phase III (i.e. a total of 873) and an optimal threshold value for the go-decision of 0.77.

04.07 (shows that req. 04.02, 04.06 and 04.11 are met):

Use the function `optimal_multiarm_binary`. Supply the same input values as in test case 04.05, however set the parameter strategy to 3, and set a constraint for the minimal success probability of 0.85.

Verify that the program returns the results for both strategies, the results for strategy 2 are the same as in test case 04.06. as the constraint is not binding, however, the results for strategy 1 change as follows: The optimal utility changes to -9999, indicating that the constraint can not be fulfilled, within the optimization region.

04.08 (shows that req. 04.01, 04.08 and 04.13 are met):

Use the function `optimal_multiarm_normal()`. Supply the following input values to the function:

- a significance level of 0.05,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.175 and 0.225,
- the optimization region of even numbers $\{10, 12, \ldots, 200\}$ for the number of participants in phase II,
- the optimization region $\{0.02, 0.04, \ldots, 0.3\}$ for the threshold values,
- boundaries of 0, 0.5 and 0.8 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 300,000,000$ and 500,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 1,500,000$ in phase II and of 2,000,000$ in phase III,
- variable costs of 67,500$ in phase II and 72,000$ in phase III,
- strategy 1, i.e. use the strategy where only the best promising treatment proceeds to phase III.

Verify that the program returns an optimal utility of 109.9 (in 10^5$), an optimal sample size of 56 in phase II and 205 in phase III (i.e. a total of 261) and an optimal threshold value for the go-decision of 0.16. Furthermore, verify, that the probability for a successful program is 0.32.

04.09 (shows that req. 04.01, 04.09 and 04.13 are met):

Use the function `optimal_multiarm_normal`. Supply the same input values as in test case 04.08, however set the parameter strategy to 2, i.e. calculating the results if all promising treatments proceed to phase III.

Verify that the program returns an optimal utility of 107.09 (in 10^5$), an optimal sample size of 30 in phase II and 247 in phase III (i.e. a total of 277) and an optimal threshold value for the go-decision of 0.20. Furthermore, verify, that the probability for a successful program is 0.33.

04.10 (shows that req. 04.01, 04.05 and 04.10 are met):

Use the function `optimal_multiarm_normal`. Supply the same input values as in test case 04.08, however set the parameter strategy to 3, i.e. calculating the results for both strategies and set a cost constraint of 200.

Verify that for strategy 1 the optimal utility changes to 109.68 (in 10^5$) and the optimal sample size changes to 46 in phase II and 190 in phase III (i.e. a total of 236). Furthermore, verify that the costs in phase II are given by 46 and the costs in phase III are given by 151. Verify that the cost constraint of 200 is returned. For strategy 2, verify that the optimal utility changes to 107.06 (in 10^5$) and the optimal sample size changes to 28 in phase II and 208 in phase III (i.e. a total of 236). Furthermore, verify that the costs in phase II are given by 34 and the costs in phase III are given by 163. Verify that the cost constraint of 200 is returned.

05. Multiple endpoints test cases

05.01 (shows that req. 05.01, 05.04, 05.11 and 05.15 are met):

Use the function `optimal_multiple_tte`. Supply the following input values to the function:

- a significance level of 0.025,
- a power of 0.9, i.e. β of 0.1,

- assumed true treatment hazard ratios of 0.75 and 0.85 for endpoint 1 and endpoint 2, respectively
- the optimization region {100, 104, ..., 300} for the number of participants in phase II,
- the optimization region {0.80, 0.82, ..., 0.9} for the threshold values,
- boundaries of 1, 0.95 and 0.85 for the effect size categories small, medium and large,
- expected gains of 100,000,000$, 150,000,000$, and 200,000,000$ for each effect size, respectively, if only endpoint 2 shows a significant result,
- expected gains of 100,000,000$, 200,000,000$, and 300,000,000$ for each effect size, respectively, if endpoint 1 shows a significant result (independent of the significance of endpoint 2),
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- a correlation between the two endpoints of 0.6,
- "fixed=FALSE", i.e. set the function to model the treatment effects on a prior distribution,
- amount of information for prior true treatment effect given by 210 and 420.

Verify that the program returns an expected utility of -212.24 (in 10^5 $) an optimal number of participants of 100 in phase II and 262 in phase III (i.e. 362 in total), and an optimal threshold value of 0.88. Furthermore, verify, that the program returns a probability to go to phase III of 0.74.

05.02 (shows that req. 05.01, 05.04, 05.05 and 05.10 are met):

Use the function `optimal_multiple_tte`. Supply the same input values as in test case 05.01, however, set a sample size constraint of 330.

Verify that the program returns an expected utility of -218.92 (in 10^5 $), an optimal number of participants of 100 in phase II and 219 in phase III (i.e. 319 in total), hence, satisfying the constraint and an optimal threshold value of 0.86.

05.03 (shows that req. 05.01, 05.03, 05.06 and 05.09 are met):

Use the function `optimal_multiple_tte`. Supply the same input values as in test case 05.01, however, set the parameter fixed to be "TRUE". Redo this, however, the second time use a maximum cost limit of 600 (in 10^5 $).

Verify that the function returns an optimal number of participants of 172 in phase II and 408 in phase III (i.e. a total of 580 participants), an optimal threshold value of 0.86 and an expected utility of 144.96 (in 10^5 $). Furthermore, verify, that the function returns costs of 229 (in 10^5 $) in phase II and 532 (in 10^5 $) in phase III. With the cost constraint, the function returns an optimal number of participants of 112 in phase II and 301 in phase III (i.e. a total of 413 participants), an optimal threshold value of 0.84 and an expected utility of 130.67 (in 10^5 $). Furthermore, verify, that the function returns costs of 184 (in 10^5 $) in phase II and 414 (in 10^5 $) in phase III, satisfying the constraint.

05.04 (shows that req. 05.01, 05.03, 05.07 and 05.12 are met):

Use the function `optimal_multiple_tte`. Supply the same input values as in test case 05.01, however, set the parameter fixed to be "TRUE" and set a minimum probability of a successful program of 0.6.

Verify that the function returns an optimal number of participants of 280 in phase II and 467 in phase III (i.e. a total of 747 participants), an optimal threshold value of 0.86 and an expected utility of 132.6 (in 10^5 $). Furthermore, verify that the probability of a successful program is given as 0.6., satisfying the constraint and that the probability that endpoint OS is significant is 0.54.

05.05 (shows that req. 05.08 is met):

Use the function `optimal_multiple_tte`. Supply the same input values as in test case 05.01, however change the number of cores for parallel computing to 6.

Verify that the computation time will increase compared to the setting in 05.01.

05.06 (shows that req. 05.02, 05.04, 05.05 and 05.10 are met):

Use the function `optimal_multiple_normal()`. Supply the following input values to the function:

- a significance level of 0.05,
- a power of 0.9, i.e. β of 0.1,
- assumed true treatment effects of 0.75 and 0.8 for the endpoints 1 and 2, respectively
  - boundaries of 0, 0.5 and 0.8 for the effect size categories small, medium and large,
- the optimization region {80, 84, ..., 160} for the number of participants in phase II,
- the optimization region {0.02, 0.04,..., 0.10} for the threshold values,
- expected gains of 100,000,000$, 200,000,000$ and 300,000,000$ for each effect size, respectively,
- twelve clusters for parallel computing,
- fixed costs of 10,000,000$ in phase II and of 15,000,000$ in phase III,
- variable costs of 75,000$ in phase II and 100,000$ in phase III,
- `fixed=FALSE`, i.e. set the function to model the treatment effects on a prior distribution,
- a correlation of 0.5 between the two treatment effects,
- variances of 2 and 1 of the two treatment effects, respectively
- sample sizes of 300 and 600 as the amount of information for the two treatment effects,
- `relaxed=TRUE`, i.e. use the relaxed combination strategy for effect sizes.

Redo this, however, the second time set a sample size constraint of 190.

Verify that the program returns an expected utility of 1476.09 (in 10^5 $), an optimal threshold value of 0.02 and optimal sample sizes of 144 in phase II and 78 in phase III (i.e. 222 in total). With the constraint, the program returns an expected utility of 809.23 (in 10^5 $), an optimal threshold value of 0.08 and optimal sample sizes of 92 in phase II and 97 in phase III (i.e. 189 in total).

05.07 (shows that req. 05.02, 05.04 and 05.08 are met):

Use the function `optimal_multiple_normal`. Supply the same input values as in test case 05.06 (without sample size constraint), however, however change the number of clusters for parallel computing to 6.

Verify that the computation time will increase compared to the setting in 05.06.

05.08 (shows that req. 05.02, 05.03, 05.06 and 05.09 are met):

Use the function `optimal_multiple_normal`. Supply the same input values as in test case 05.06 (without sample size constraint), however, set the parameter fixed to be "TRUE". Redo this, however, the second time use a maximum cost limit of 400 (in 10^5 $).

Verify that the program returns an expected utility of 1133 (in 10^5 $), an optimal threshold value of 0.02, an optimal sample size in phase II of 160. Furthermore verify, that the costs are 220 (in 10^5 $) in phase II and 215 (in 10^5 $) in phase III, i.e. a total of 407 (in 10^5 $). Verify, that due to the cost constraint the program now returns an expected utility of 1099 (in 10^5 $), an optimal threshold value of 0.02, an optimal sample size in phase II of 104. Furthermore verify, that the costs are 178 (in 10^5 $) in phase II and 220 (in 10^5 $) in phase III, i.e. a total of 398 (in 10^5 $), thus, satisfying the constraint.

05.09 (shows that req. 05.02, 05.03, 05.07 and 05.12 are met):

Use the function `optimal_multiple_normal`. Supply the same input values as in test case 05.06 (without sample size constraint), however, set the parameter fixed to be "TRUE". Redo this and set a minimum probability of a successful program of 0.85.

Verify that the program returns an expected utility of 1133 (in 10^5 $), an optimal threshold value of 0.02, an optimal sample size in phase II of 160. Furthermore, verify the probability of a successful program is 0.835, and the success probabilities for the various benefit categories are given by 0.15, 0.637 and 0.048 for small, medium and large treatment effects respectively. Furthermore verify, that the constraint cannot be met, within the optimization region, i.e. an expected utility of -9999 is returned, when the constraint is imposed.

05.10 (shows that req. 05.11, 05.13 and 05.14 are met):

Use the function `optimal_multiple_normal`. Supply the same input values as in test case 05.06 (without sample size constraint), however, set parameter fixed to be "TRUE". Redo this, however the second time, set the parameter relaxed to "FALSE"

Verify that for relaxed = "TRUE" the program returns an expected utility of 1133 (in 10^5 $), an optimal threshold value of 0.02, an optimal sample size in phase II of 160 and a probability to go to phase III of 0.99. For relaxed = "FALSE", the program returns an expected utility of 410 (in 10^5 $), an optimal threshold value of 0.02, an optimal sample size in phase II of 132 and a probability to go to phase III of 0.98.

Test Results

Test	Results	Pass/Fail
01.01.1	As expected	Pass
01.01.2	As expected	Pass
01.01.3	As expected	Pass
01.01.4	As expected	Pass
01.01.5	As expected	Pass
01.01.6	As expected	Pass
01.01.7	As expected	Pass
01.01.8	As expected	Pass
01.02.1	As expected	Pass
01.02.2	As expected	Pass
01.02.3	As expected	Pass
01.02.4	As expected	Pass
01.02.5	As expected	Pass
01.02.6	As expected	Pass
01.02.7	As expected	Pass
01.03.1	As expected	Pass
01.03.2	As expected	Pass
01.03.3	As expected	Pass
01.03.4	As expected	Pass
01.03.5	As expected	Pass
01.04.1	As expected	Pass
01.04.2	As expected	Pass
01.04.3	As expected	Pass
01.04.4	As expected	Pass
01.04.5	As expected	Pass
01.05.1	As expected	Pass
01.05.2	As expected	Pass
01.05.3	As expected	Pass
01.05.4	As expected	Pass
01.05.5	As expected	Pass
01.06.1	As expected	Pass
01.06.2	As expected	Pass
01.06.3	As expected	Pass
01.07.1	As expected	Pass
01.07.2	As expected	Pass
01.07.3	As expected	Pass
01.08.1	As expected	Pass
01.08.2	As expected	Pass

01.08.3	As expected	Pass
01.08.4	As expected	Pass
01.08.5	As expected	Pass
01.08.6	As expected	Pass
01.09.1	As expected	Pass
01.09.2	As expected	Pass
01.09.3	As expected	Pass
01.10.1	As expected	Pass
01.10.2	As expected	Pass
01.10.3	As expected	Pass
01.11.1	As expected	Pass
01.11.2	As expected	Pass
01.11.3	As expected	Pass
01.11.4	As expected	Pass
01.11.5	As expected	Pass
01.11.6	As expected	Pass
01.11.7	As expected	Pass
01.12.1	As expected	Pass
01.13.1	As expected	Pass
01.13.2	As expected	Pass
01.14.1	As expected	Pass
01.14.2	As expected	Pass
01.15.1	As expected	Pass
01.15.2	As expected	Pass

Test	Results	Pass/Fail
02.01.1	As expected	Pass
02.01.2	As expected	Pass
02.01.3	As expected	Pass
02.01.4	As expected	Pass
02.01.5	As expected	Pass
02.01.6	As expected	Pass
02.01.7	As expected	Pass
02.01.8	As expected	Pass
02.02.1	As expected	Pass
02.02.2	As expected	Pass
02.02.3	As expected	Pass
02.02.4	As expected	Pass
02.02.5	As expected	Pass
02.02.6	As expected	Pass
02.03.1	As expected	Pass
02.03.2	As expected	Pass
02.03.3	As expected	Pass
02.03.4	As expected	Pass
02.03.5	As expected	Pass
02.03.6	As expected	Pass
02.03.7	As expected	Pass
02.03.8	As expected	Pass
02.03.9	As expected	Pass
02.03.10	As expected	Pass

02.03.11	As expected	Pass
02.03.12	As expected	Pass
02.03.13	As expected	Pass
02.04.1	As expected	Pass
02.04.2	As expected	Pass
02.04.3	As expected	Pass
02.04.4	As expected	Pass
02.04.5	As expected	Pass
02.04.6	As expected	Pass
02.04.7	As expected	Pass
02.04.8	As expected	Pass
02.04.9	As expected	Pass
02.04.10	As expected	Pass
02.04.11	As expected	Pass
02.04.12	As expected	Pass
02.04.13	As expected	Pass
02.04.14	As expected	Pass
02.04.15	As expected	Pass
02.04.16	As expected	Pass
02.04.17	As expected	Pass
02.04.18	As expected	Pass
02.04.19	As expected	Pass
02.04.20	As expected	Pass
02.04.21	As expected	Pass
02.04.22	As expected	Pass
02.04.23	As expected	Pass
02.05.1	As expected	Pass
02.05.2	As expected	Pass
02.05.3	As expected	Pass
02.05.4	As expected	Pass
02.05.5	As expected	Pass
02.05.6	As expected	Pass
02.05.7	As expected	Pass
02.05.8	As expected	Pass
02.05.9	As expected	Pass
02.05.10	As expected	Pass
02.05.11	As expected	Pass
02.05.12	As expected	Pass
02.06.1	As expected	Pass
02.06.2	As expected	Pass
02.06.3	As expected	Pass
02.06.4	As expected	Pass
02.06.5	As expected	Pass
02.06.6	As expected	Pass
02.06.7	As expected	Pass
02.06.8	As expected	Pass
02.06.9	As expected	Pass
02.06.10	As expected	Pass
02.06.11	As expected	Pass
02.06.12	As expected	Pass
02.06.13	As expected	Pass

02.06.14	As expected	Pass
02.06.15	As expected	Pass
02.06.16	As expected	Pass
02.07.1	As expected	Pass
02.07.2	As expected	Pass
02.07.3	As expected	Pass
02.07.4	As expected	Pass
02.07.5	As expected	Pass
02.07.6	As expected	Pass
02.07.7	As expected	Pass
02.07.8	As expected	Pass
02.07.9	As expected	Pass
02.07.10	As expected	Pass
02.07.11	As expected	Pass
02.07.12	As expected	Pass
02.09.1	As expected	Pass
02.09.2	As expected	Pass
02.09.3	As expected	Pass
02.09.4	As expected	Pass
02.09.5	As expected	Pass
02.10.1	As expected	Pass
02.10.2	As expected	Pass
02.10.3	As expected	Pass
02.10.4	As expected	Pass
02.10.5	As expected	Pass
02.10.6	As expected	Pass
02.10.7	As expected	Pass
02.10.8	As expected	Pass
02.10.9	As expected	Pass
02.10.10	As expected	Pass
02.10.11	As expected	Pass
02.10.12	As expected	Pass
02.10.13	As expected	Pass
02.10.14	As expected	Pass
02.10.15	As expected	Pass
02.10.16	As expected	Pass
02.10.17	As expected	Pass
02.10.18	As expected	Pass
02.10.19	As expected	Pass
02.10.20	As expected	Pass
02.10.21	As expected	Pass
02.11.1	As expected	Pass
02.11.2	As expected	Pass
02.11.3	As expected	Pass
02.11.4	As expected	Pass
02.11.5	As expected	Pass
02.11.6	As expected	Pass
02.12.1	As expected	Pass
02.12.2	As expected	Pass
02.12.3	As expected	Pass
02.12.4	As expected	Pass

02.12.5	As expected	Pass
02.12.6	As expected	Pass
02.12.7	As expected	Pass
02.12.8	As expected	Pass
02.12.9	As expected	Pass
02.12.10	As expected	Pass
02.12.11	As expected	Pass
02.12.12	As expected	Pass
02.12.13	As expected	Pass
02.12.14	As expected	Pass

Test	Results	Pass/Fail
03.01.1	As expected	Pass
03.01.2	As expected	Pass
03.01.3	As expected	Pass
03.01.4	As expected	Pass
03.01.5	As expected	Pass
03.01.6	As expected	Pass
03.01.7	As expected	Pass
03.01.8	As expected	Pass
03.01.9	As expected	Pass
03.01.10	As expected	Pass
03.01.11	As expected	Pass
03.01.12	As expected	Pass
03.01.13	As expected	Pass
03.01.14	As expected	Pass
03.01.15	As expected	Pass
03.01.16	As expected	Pass
03.01.17	As expected	Pass
03.01.18	As expected	Pass
03.01.19	As expected	Pass
03.01.20	As expected	Pass
03.01.21	As expected	Pass
03.01.22	As expected	Pass
03.01.23	As expected	Pass
03.01.24	As expected	Pass
03.01.25	As expected	Pass
03.01.26	As expected	Pass
03.01.27	As expected	Pass
03.01.28	As expected	Pass
03.01.29	As expected	Pass
03.01.30	As expected	Pass
03.01.31	As expected	Pass
03.01.32	As expected	Pass
03.01.33	As expected	Pass
03.02.1	As expected	Pass
03.02.2	As expected	Pass
03.02.3	As expected	Pass
03.02.4	As expected	Pass
03.02.5	As expected	Pass

03.02.6	As expected	Pass
03.02.7	As expected	Pass
03.03.1	As expected	Pass
03.03.2	As expected	Pass
03.03.3	As expected	Pass
03.03.4	As expected	Pass
03.03.5	As expected	Pass
03.03.6	As expected	Pass
03.03.7	As expected	Pass
03.03.8	As expected	Pass
03.03.9	As expected	Pass
03.03.10	As expected	Pass
03.03.11	As expected	Pass
03.03.12	As expected	Pass
03.03.13	As expected	Pass
03.03.14	As expected	Pass
03.03.15	As expected	Pass
03.03.16	As expected	Pass
03.03.17	As expected	Pass
03.03.18	As expected	Pass
03.03.19	As expected	Pass
03.03.20	As expected	Pass
03.03.21	As expected	Pass
03.03.22	As expected	Pass
03.03.23	As expected	Pass
03.03.24	As expected	Pass
03.03.25	As expected	Pass
03.03.26	As expected	Pass
03.03.27	As expected	Pass
03.03.28	As expected	Pass
03.03.29	As expected	Pass
03.03.30	As expected	Pass
03.03.31	As expected	Pass
03.04.1	As expected	Pass
03.05.1	As expected	Pass
03.05.2	As expected	Pass
03.05.3	As expected	Pass
03.05.4	As expected	Pass
03.05.5	As expected	Pass
03.05.6	As expected	Pass
03.05.7	As expected	Pass
03.05.8	As expected	Pass
03.05.9	As expected	Pass
03.05.10	As expected	Pass
03.05.11	As expected	Pass
03.05.12	As expected	Pass
03.05.13	As expected	Pass
03.05.14	As expected	Pass
03.05.15	As expected	Pass
03.05.16	As expected	Pass
03.05.17	As expected	Pass

03.05.18	As expected	Pass
03.05.19	As expected	Pass
03.05.20	As expected	Pass
03.05.21	As expected	Pass
03.05.22	As expected	Pass
03.05.23	As expected	Pass
03.05.24	As expected	Pass
03.05.25	As expected	Pass
03.05.26	As expected	Pass
03.05.27	As expected	Pass
03.05.28	As expected	Pass
03.05.29	As expected	Pass
03.05.30	As expected	Pass
03.05.31	As expected	Pass
03.05.32	As expected	Pass
03.05.33	As expected	Pass
03.06.1	As expected	Pass
03.06.2	As expected	Pass
03.06.3	As expected	Pass
03.06.4	As expected	Pass
03.06.5	As expected	Pass
03.06.6	As expected	Pass
03.06.7	As expected	Pass
03.06.8	As expected	Pass
03.06.9	As expected	Pass
03.07.1	As expected	Pass
03.07.2	As expected	Pass
03.07.3	As expected	Pass
03.07.4	As expected	Pass
03.07.5	As expected	Pass
03.07.6	As expected	Pass
03.07.7	As expected	Pass
03.07.8	As expected	Pass
03.07.9	As expected	Pass
03.07.10	As expected	Pass
03.07.11	As expected	Pass
03.07.12	As expected	Pass
03.07.13	As expected	Pass
03.07.14	As expected	Pass
03.07.15	As expected	Pass
03.07.16	As expected	Pass
03.08.1	As expected	Pass
03.08.2	As expected	Pass
03.08.3	As expected	Pass
03.08.4	As expected	Pass
03.09.1	As expected	Pass
03.09.2	As expected	Pass
03.09.3	As expected	Pass
03.09.4	As expected	Pass
03.09.5	As expected	Pass
03.09.6	As expected	Pass

03.09.7	As expected	Pass
03.09.8	As expected	Pass
03.09.9	As expected	Pass
03.09.10	As expected	Pass
03.09.11	As expected	Pass
03.10.1	As expected	Pass
03.10.2	As expected	Pass
03.10.3	As expected	Pass
03.10.4	As expected	Pass
03.10.5	As expected	Pass
03.10.6	As expected	Pass
03.10.7	As expected	Pass
03.10.8	As expected	Pass
03.10.9	As expected	Pass
03.11.1	As expected	Pass
03.11.2	As expected	Pass
03.11.3	As expected	Pass
03.11.4	As expected	Pass
03.11.5	As expected	Pass
03.11.6	As expected	Pass
03.11.7	As expected	Pass
03.11.8	As expected	Pass
03.11.9	As expected	Pass
03.11.10	As expected	Pass
03.11.11	As expected	Pass
03.11.12	As expected	Pass
03.11.13	As expected	Pass
03.11.14	As expected	Pass
03.11.15	As expected	Pass
03.11.16	As expected	Pass
03.11.17	As expected	Pass
03.11.18	As expected	Pass
03.11.19	As expected	Pass
03.12.1	As expected	Pass
03.13.1	As expected	Pass
03.13.2	As expected	Pass
03.13.3	As expected	Pass
03.13.4	As expected	Pass
03.13.5	As expected	Pass
03.13.6	As expected	Pass
03.13.7	As expected	Pass
03.13.8	As expected	Pass
03.13.9	As expected	Pass
03.13.10	As expected	Pass
03.13.11	As expected	Pass
03.13.12	As expected	Pass
03.13.13	As expected	Pass
03.13.14	As expected	Pass
03.13.15	As expected	Pass
03.13.16	As expected	Pass
03.13.17	As expected	Pass

03.13.18	As expected	Pass
03.13.19	As expected	Pass
03.13.20	As expected	Pass
03.13.21	As expected	Pass
03.13.22	As expected	Pass
03.13.23	As expected	Pass
03.13.24	As expected	Pass
03.13.25	As expected	Pass
03.13.26	As expected	Pass
03.13.27	As expected	Pass
03.13.28	As expected	Pass
03.13.29	As expected	Pass
03.13.30	As expected	Pass
03.13.31	As expected	Pass
03.13.32	As expected	Pass
03.13.33	As expected	Pass
03.14.1	As expected	Pass
03.14.2	As expected	Pass
03.14.3	As expected	Pass
03.14.4	As expected	Pass
03.14.5	As expected	Pass

Test	Results	Pass/Fail
04.01.1	As expected	Pass
04.01.2	As expected	Pass
04.01.3	As expected	Pass
04.01.4	As expected	Pass
04.01.5	As expected	Pass
04.01.6	As expected	Pass
04.02.1	As expected	Pass
04.02.2	As expected	Pass
04.02.3	As expected	Pass
04.02.4	As expected	Pass
04.02.5	As expected	Pass
04.02.6	As expected	Pass
04.03.1	As expected	Pass
04.03.2	As expected	Pass
04.03.3	As expected	Pass
04.03.4	As expected	Pass
04.03.5	As expected	Pass
04.03.6	As expected	Pass
04.03.7	As expected	Pass
04.03.8	As expected	Pass
04.03.9	As expected	Pass
04.03.10	As expected	Pass
04.03.11	As expected	Pass
04.04.1	As expected	Pass
04.05.1	As expected	Pass
04.05.2	As expected	Pass
04.05.3	As expected	Pass

04.05.4	As expected	Pass
04.05.5	As expected	Pass
04.06.1	As expected	Pass
04.06.2	As expected	Pass
04.06.3	As expected	Pass
04.06.4	As expected	Pass
04.06.5	As expected	Pass
04.07.1	As expected	Pass
04.07.2	As expected	Pass
04.07.3	As expected	Pass
04.07.4	As expected	Pass
04.07.5	As expected	Pass
04.07.6	As expected	Pass
04.08.1	As expected	Pass
04.08.2	As expected	Pass
04.08.3	As expected	Pass
04.08.4	As expected	Pass
04.08.5	As expected	Pass
04.08.6	As expected	Pass
04.09.1	As expected	Pass
04.09.2	As expected	Pass
04.09.3	As expected	Pass
04.09.4	As expected	Pass
04.09.5	As expected	Pass
04.09.6	As expected	Pass
04.10.1	As expected	Pass
04.10.2	As expected	Pass
04.10.3	As expected	Pass
04.10.4	As expected	Pass
04.10.5	As expected	Pass
04.10.6	As expected	Pass
04.10.7	As expected	Pass
04.10.8	As expected	Pass
04.10.9	As expected	Pass
04.10.10	As expected	Pass
04.10.11	As expected	Pass
04.10.12	As expected	Pass
04.10.13	As expected	Pass
04.10.14	As expected	Pass

Test	Results	Pass/Fail
05.01.1	As expected	Pass
05.01.2	As expected	Pass
05.01.3	As expected	Pass
05.01.4	As expected	Pass
05.01.5	As expected	Pass
05.01.6	As expected	Pass
05.02.1	As expected	Pass
05.02.2	As expected	Pass
05.02.3	As expected	Pass

05.02.4	As expected	Pass
05.02.5	As expected	Pass
05.03.1	As expected	Pass
05.03.2	As expected	Pass
05.03.3	As expected	Pass
05.03.4	As expected	Pass
05.03.5	As expected	Pass
05.03.6	As expected	Pass
05.03.7	As expected	Pass
05.03.8	As expected	Pass
05.03.9	As expected	Pass
05.03.10	As expected	Pass
05.03.11	As expected	Pass
05.03.12	As expected	Pass
05.03.13	As expected	Pass
05.03.14	As expected	Pass
05.04.1	As expected	Pass
05.04.2	As expected	Pass
05.04.3	As expected	Pass
05.04.4	As expected	Pass
05.04.5	As expected	Pass
05.04.6	As expected	Pass
05.04.7	As expected	Pass
05.05.1	As expected	Pass
05.06.1	As expected	Pass
05.06.2	As expected	Pass
05.06.3	As expected	Pass
05.06.4	As expected	Pass
05.06.5	As expected	Pass
05.06.6	As expected	Pass
05.06.7	As expected	Pass
05.06.8	As expected	Pass
05.06.9	As expected	Pass
05.06.10	As expected	Pass
05.07.1	As expected	Pass
05.08.1	As expected	Pass
05.08.2	As expected	Pass
05.08.3	As expected	Pass
05.08.4	As expected	Pass
05.08.5	As expected	Pass
05.08.6	As expected	Pass
05.08.7	As expected	Pass
05.08.8	As expected	Pass
05.08.9	As expected	Pass
05.08.10	As expected	Pass
05.09.1	As expected	Pass
05.09.2	As expected	Pass
05.09.3	As expected	Pass
05.09.4	As expected	Pass
05.09.5	As expected	Pass
05.09.6	As expected	Pass

05.09.7	As expected	Pass
05.09.8	As expected	Pass
05.10.1	As expected	Pass
05.10.2	As expected	Pass
05.10.3	As expected	Pass
05.10.4	As expected	Pass
05.10.5	As expected	Pass
05.10.6	As expected	Pass
05.10.7	As expected	Pass
05.10.8	As expected	Pass

Table 10: Summary of failures and passes

Setting	Failures	Passes
01 Basic	0	62
02 Bias	0	136
03 Multitrial	0	212
04 Multiarm	0	66
05 Multiple	0	70
Total	0	546

Table 11: Duration of test code runs

Setting	Duration
01 Basic	0.221 hours
02 Bias	1.2 hours
03 Multitrial	11.5 hours
04 Multiarm	3.36 hours
05 Multiple	16.8 hours
Total	33 hours

References

Götte, Heiko, Armin Schüler, Marietta Kirchner, and Meinhard Kieser. 2015. "Sample Size Planning for Phase II Trials Based on Success Probabilities for Phase III." *Pharmaceutical Statistics* 14 (6): 515–24. https://doi.org/10.1002/pst.1717.

Hughes, E. 2021. "White Paper WP-059: R Package Validation Framework." PHUSE Working Group: Data Visualisation & Open Source Technology. 2021. https://advance.phuse.global/display/WEL/R+Package+Validation+Framework.

Hughes, E., Vendettuoli, M., Miller, E., Peyman, and E. 2021. *Automate Validated Package Creation.* https://github.com/phuse-org/valtools.

Kieser, Meinhard, Marietta Kirchner, Eva Dölger, and Heiko Götte. 2018. "Optimal Planning of Phase II/III Programs for Clinical Trials with Multiple Endpoints." *Pharmaceutical Statistics* 17 (5): 437–57. https://doi.org/10.1002/pst.1861.

Preussler, Stella. 2020. *Integrated Planning of Pilot and Subsequent Confirmatory Study in Clinical Research: Finding Optimal Designs in a Utility-Based Framework.* Heidelberg: Dissertation, supervised by Kieser, M.