Keaven M. Anderson

gsDesign Technical Manual

Placeholder Name

gsDesign Technical Manual

© Placeholder Name, Inc.

ISBN-1234567891234

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et elementum libero. In hac habitasse platea dictumst. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur blandit tempus porttitor. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.

Placeholder for dedication text

Table of contents

Welcome

Welcome to the **gsDesign Technical Manual** by Keaven M. Anderson. You can view the HTML version at https://keaven.github.io/gsd-tech-manual/.

The gsDesign package supports group sequential clinical trial design. While there is a strong focus on designs using α- and β-spending functions, Wang-Tsiatis designs, including O'Brien-Fleming and Pocock designs, are also available. The ability to design with non-binding futility rules is an important feature to control Type I error in a manner acceptable to regulatory authorities.

The book was written by Keaven M. Anderson. The translation from the original LaTeX version to bookdown was largely done by Fang Xiao, with help from Yilong Zhang and Nan Xiao. Later, Nan Xiao migrated the content from bookdown to Quarto.

Preface

The gsDesign package supports group sequential clinical trial design. While there is a strong focus on designs using α- and β-spending functions, Wang-Tsiatis designs, including O'Brien-Fleming and Pocock designs, are also available. The ability to design with non-binding futility rules is an important feature to control Type I error in a manner acceptable to regulatory authorities.

The routines are designed to provide simple access to commonly used designs using default arguments. Standard, published spending functions are supported as well as the ability to write custom spending functions. A `gsDesign` class is defined and returned by the `gsDesign()` function. A plot function for this class provides a wide variety of plots: boundaries, power, estimated treatment effect at boundaries, conditional power at boundaries, spending function plots, expected sample size plot, and B-values at boundaries. Using function calls to access the package routines provides a powerful capability to derive designs or output formatting that could not be anticipated through a GUI interface. However, basic functionality is provided with the Shiny interface at https://rinpharma.shinyapps.io/gsdesign/. This enables the user to easily create designs with features they desire, such as designs with minimum expected sample size.

In addition to straightforward group sequential design, the gsDesign package provides tools to effectively adapt clinical trials during execution. First, the spending function approach to design allows altering timing of analyses during the course of the trial. Information-based timing of analyses allows adaptation of sample size or number of events to ensure adequate power for a trial. Finally, gsDesign provides a routine that enable design adaptation using conditional error and conditional power.

In summary, the intent of the gsDesign package is to easily create, fully characterize, and even optimize routine group sequential trial designs, as well as to provide a tool to derive and evaluate innovative designs.

Version history

Version 2.2 adds high-quality plots using the ggplot2 package and additional calculations available for Bayesian calculation such as predictive power and computing probability of success by averaging power over a prior distribution for treatment effect. A GUI interface is also available through the gsDesign-Explorer R package that is available separately (not on CRAN); a separate manual is also available.

Version 2.3 provides boundary summary functions `gsBoundSummary()` and `xtable.gsDesign()`. The provide many summary values for design boundaries for on-screen (`gsBoundSummary()`) and LaTeX (`xtable.gsDesign()`) output. Fixes for plotting for one-sided designs are also made in this version.

Version 2.4 adds functions and plots summarizing treatment effect approximations based on interim Z-statistics.

Version 2.5 adds posterior distribution computation for the parameter of interest, θ, as well as prediction intervals.

Upper Gwynedd, Pennsylvania Keaven M. Anderson

Chapter 1

Introduction

1.1 Overview

The gsDesign package is intended to provide a flexible set of tools for designing and analyzing group sequential trials. There are other adaptive methods that can also be supported using this underlying toolset. This manual is intended as an introduction to gsDesign. Many users may just want to apply basic, standard design methods. Others will be interested in applying the toolset to very ambitious adaptive designs. We try to give some orientation to each of these sets of users, and to distinguish between the material needed by each. For those looking for a particularly simple approach to using gsDesign, a web-based Shiny app is available at https://rinpharma.shinyapps.io/gsdesign/.

The remainder of this overview provides a quick review of topics covered in this manual. The introduction continues with some basic theory behind group sequential design to provide background for the routines. There is no attempt to fully develop the theory for statistical tests or for group sequential design in general since many statisticians will already be familiar with these and there are excellent texts available such as Jennison and Turnbull [2000] and Proschan et al. [2006].

The introduction continues with a simple outline of the main routines provided in the gsDesign package followed by motivational examples that will be used later in the manual. Basic sample size calculations for 2-arm binomial outcome trials using the `nBinomial()` function and 2-arm time-to-event endpoint trials using `nSurvival()` are shown, including an example of a noninferiority trial. Both superiority and noninferiority trials are considered.

Further material is arranged by topic in subsequent sections. Chapter 2 provides a minimal background in asymptotic probability theory for group sequential testing. The basic calculations involve computing boundary crossing probabilities for correlated normal random variables. We demonstrate the `gsProbability()` routine to compute boundary crossing probabilities and expected sample size for group sequential designs.

Setting boundaries for group sequential designs, particularly using spending functions is the main point of emphasis in the gsDesign package. Section 4.1 through Section 8.5 of the manual present the design and evaluation of designs for group sequential trials using the `gsDesign()` routine.

Default parameters for `gsDesign()` are demonstrated for the motivational examples in Section 4.1. Basic computations for group sequential designs using boundary families and error spending are provided in Chapter 6. The primary discussion of Wang-Tsiatis boundary families [Wang and Tsiatis, 1987] (e.g., O'Brien-Fleming O'Brien and Fleming [1979] and Pocock Pocock [1977] designs) is provided here in Section 6.2.

Next, we proceed to a short discussion in Chapter 7 of `gsDesign()` parameters for setting Type I and II error rates and the number and timing of analyses. The section also explains how to use a measure of treatment effect to size trials, with specific discussion of event-based computations for trials with time-to-event analyses.

The basics of standard spending functions are provided in Chapter 8. Subsections defining spending functions and spending function families are followed by a description of how to use built-in standard Hwang-Shih-DeCani Hwang et al. [1990] and power Kim and Demets [1987] spending functions in `gsDesign()`. Section 8.4 shows how to reset timing of interim analyses using `gsDesign()`.

The final section on spending functions is Section 8.5 which presents details of how spending functions are defined for `gsDesign()` and other advanced topics that will probably not be needed by many users. The section will be of use to those interested in investigational spending functions and optimized spending function choice. Spending function families by Anderson and Clark [2010] providing additional flexibility to standard one-parameter spending functions are detailed as part of a comprehensive list of built-in spending functions. This is followed by examples of how to derive optimal designs and how to implement new spending functions.

Next comes Section Chapter 9 on the basic analysis of group sequential trials. This includes computing stagewise and repeated p-values as well as repeated confidence intervals.

Conditional power and B-values are presented in Chapter 10. These are methods used for evaluating interim trends in a group sequential design, but may also be used to adapt a trial design at an interim analysis using the methods of Müller and Schäfer [2001]. The routine `gsCP()` provides the basis for applying these adaptive methods.

We end with a discussion of Bayesian computations in Chapter 11. The gsDesign package can quite simply be used with decision theoretic methods to derive optimal designs. We also apply Bayesian computations to update the probability of success a trial based on knowing a bound has not been crossed, but without knowledge of unblinded treatment results.

Future extensions of the manual could further discuss implementation of information-based designs and additional adaptive design topics.

1.2 Quick start: installation and online help

This brief section is meant to get you up and going. The package is most easily downloaded and installed in R from the CRAN website using:

```r
install.packages("gsDesign")
```

Since the package is released on a regular basis, there is usually little value installing the most recent development version from GitHub using:

```r
remotes::install_github("keaven/gsDesign")
```

After installation, attach the gsDesign package with:

```r
library(gsDesign)
```

Online help can be obtained by entering the following on the command line:

```r
help(gsDesign)
```

There are many help topics covered there which should be sufficient information to keep you from needing to use this document for day-to-day use or if you just generally prefer not using a manual. The same information plus additional vignettes providing more long-form description of usage are at https://keaven.github.io/gsDesign/.

1.3 Package testing

While there are no guarantees provided, we note that extensive unit testing has been written to ensure package quality. At the time of this writing, code coverage is at 82%.

1.4 The primary routines in the gsDesign package

As an overview to the R package, there is a handful of R functions that provide basic computations related to designing and evaluating many group sequential clinical trials:

1. The `gsDesign()` function provides sample size and boundaries for a group sequential design based on treatment effect, spending functions for boundary crossing probabilities, and relative timing of each analysis. Standard and user-specified spending functions may be used. In addition to spending function designs, the family of Wang-Tsiatis designs—including O'Brien-Fleming and Pocock designs—are also available.
2. The `gsSurv()` function extends the `gsDesign()` function to design group sequential trials for time-to-event endpoints.
3. The `gsProbability()` function computes boundary crossing probabilities and expected sample size of a design for arbitrary user-specified treatment effects, bounds, and interim analysis sample sizes.
4. The `gsCP()` function computes the conditional probability of future boundary crossing given a result at an interim analysis. There are related functions to support other conditional and predictive power calculations that can be used for interim descriptions of or adaptive design. For instance, the `ssrCP()` function supports sample size re-estimation directly for 2-stage trials.

We note that the full reference of all functions is organized by topic and easily available at the package documentation site (https://keaven.github.io/gsDesign/) in the reference section.

The package design strategy should make its tools useful both as an everyday tool for simple group sequential design as well as a research tool for a wide variety of group sequential design problems. Both `print()` and `plot()` functions are available for both `gsDesign()` and `gsProbability()`. The `gsBoundSummary()` function provides a formatted table for incorporation in a protocol; `gsBoundSummary()` provides the capability to summarize many boundary characteristics, including conditional power, treatment effect approximations and B-values. This is particularly helpful with the gt package to publish tables.

The most extensive set of supportive routines enables design and evaluation of binomial endpoint trials and time-to-event endpoint trials. For binomial endpoints, we use the Farrington and Manning [1990] method for sample size estimation in `nBinomial()` and the corresponding Miettinen and Nurminen [1985] method for testing, confidence intervals, and simulation. We also implement the Lachin and Foulkes [1986] for sample size for survival studies. The examples we present apply these methods to group sequential trial design for binomial and time-to-event endpoints.

Functions are set up to be called directly from the R command line. Default arguments and output for `gsDesign()` and `gsSurv()` are included to make initial use simple. Sufficient options are available, however, to make the routine very flexible. Guided use not requiring looking up function arguments is provided by the Shiny interface at https://rinpharma.shinyapps.io/gsdesign/.

Simple examples provide the best overall motivation for group sequential design. This manual does not attempt to comprehensively delineate all that the gsDesign package may accomplish. The intent is to include enough detail to demonstrate a variety of approaches to group sequential design that provide the user with a useful tool and the understanding of ways that it may be applied and extended. Examples that will reappear throughout the manual are introduced here.

1.5 The CAPTURE trial: binary endpoint example

The CAPTURE Investigators [1997] presented the results of a randomized trial in patients with unstable angina who required treatment with angioplasty, an invasive procedure where a balloon is inflated in one or more coronary arteries to reduce blockages. In the process of opening a coronary artery, the balloon can injure the artery which may lead to thrombotic complications. Standard treatment at the time the trial was run included treatment with heparin and aspirin before and during angioplasty to reduce the thrombotic complications such as the primary composite endpoint comprising myocardial infarction, recurrent urgent coronary intervention and death over the course of 30 days. This trial compared this standard therapy to the same therapy plus abciximab, a platelet inhibitor. While the original primary analysis used a logrank statistic to compare treatment groups, for this presentation we will consider the outcome binary. Approximately 15% of patients in the control group were expected to experience a primary endpoint, but rates from 7.5% to 20% could not be ruled out. There was an expectation that the experimental treatment would reduce incidence of the primary endpoint by at least 1/3, but possibly by as much as 1/2 or 2/3. Since a 1/3 reduction was felt to be conservative, the trial was planned to have 80% power. Given these various possibilities, the desirable sample size for a trial with a fixed design had over a 10-fold range from 202 to 2942; see Table below.

```r
n <- NULL
p <- c(0.075, 0.1, 0.15, 0.2)
for (p1 in p) {
  n <- rbind(
    n,
    ceiling(
      nBinomial(
        p1 = p1,
        p2 = p1 * c(2 / 3, 1 / 2, 1 / 3),
        beta = 0.2
      ) / 2
    ) * 2
  )
}
tb <- data.frame(p * 100, n)
names(tb) <- c(
  "Control rate (%)",
  "1/3 reduction",
  "1/2 reduction",
  "2/3 reduction"
)
tb %>%
  kable(
    caption = paste0(
      "Fixed design sample size possibilities ",
      "for the CAPTURE trial by control group event rate ",
      "and relative treatment effect."
    )
  ) %>%
  kable_styling()
```

Table 1.1: Fixed design sample size possibilities for the CAPTURE trial by control group event rate and relative treatment effect.

Control rate (%)	1/3 reduction	1/2 reduction	2/3 reduction
7.5	2942	1184	596
10.0	2158	870	438
15.0	1372	556	282
20.0	980	398	202

The third line in the above table can be generated using the call

```
nBinomial(
  p1 = 0.15,
  p2 = 0.15 * c(2 / 3, 1 / 2, 1 / 3),
  beta = 0.2
)
#> [1] 1371.1937  554.9067  280.1902
```

and rounding the results up to the nearest even number. The function `nBinomial()` in the gsDesign package is designed to be a flexible tool for deriving sample size for two-arm binomial trials for both superiority and non-inferiority. Type at the command prompt to see background on sample size, simulation, testing and confidence interval routines for fixed (non-group sequential) binomial trials. These routines will be used with this and other examples throughout the manual.

1.6 A time-to-event endpoint in a cancer trial

As a second example we consider comparing a new treatment to a standard treatment for a cancer trial. Lachin and Foulkes [Lachin and Foulkes, 1986] provide a method of computing sample size assuming the following distributions are known:

- The time to a primary endpoint in each treatment group.
- The time until dropout in each group.
- Enrollment over time.

Statistical testing is performed using the logrank test statistic. The methods allow different assumptions in different strata. Enrollment time and total study duration are assumed fixed, and the sample size and number of events required during those periods, respectively, to achieve a desired power and Type I error are computed. Here we apply the simplest form of this method, assuming an exponential distribution in each case with no stratification. The routine can be used to derive the sample size and number of events required. This routine works with failure rates rather than distribution medians or dropout rates per year. An exponential distribution with failure rate λ has cumulative probability of failure at or before time t of

$$F(t) = 1 - e^{-\lambda t}.$$

If the cumulative failure rate is known to be p_0 at time t_0, then the value of λ is

$$\lambda = -\ln(1 - p_0)/t_0.$$

We assume for the trial of interest that the primary endpoint is the time from randomization until the first of disease progression or death (progression free survival or PFS). Patients on the standard treatment are assumed to have an exponential failure rate with a median PFS of 6 months, yielding $\lambda_C = \ln(2)/6 = 0.1155$ with t measured in months. The trial is to be powered at 90% to detect a reduction in the hazard rate for PFS of 30% (HR = 0.7) in the experimental group compared to standard treatment. This yields an experimental group failure rate of $0.7 \times \lambda_C = 0.0809$. Patients are assumed to drop out at a rate of 5% per year of follow-up which implies an exponential rate $\eta = -\ln(0.95)/12 = 0.00427$. Enrollment is assumed to be uniform over 30 months with patients followed for a minimum of 6 months, yielding a total study time of 36 months.

The function is `nSurv()` computes sample size using the Lachin and Foulkes [Lachin and Foulkes, 1986] method:

```r
x <- nSurv(
  lambdaC = log(2) / 6,
  alpha = 0.025,
  beta = 0.1,
  eta = -log(0.95) / 12,
  hr = 0.7,
  T = 36,
  minfup = 6
)
```

This returns a total sample size `x$n` of 416.2635478 which is a continuous number. Generally, you will want to round up to an even number with

```r
n <- ceiling(x$n / 2) * 2
n
```

The target number of events to power the trial is rounded up to the nearest integer:

```r
events <- ceiling(x$d)
events
#> [1] 330
```

Thus, 2942, 2158, 1372, 980, 1184, 870, 556, 398, 596, 438, 282, 202 patients and 330 events are sufficient to obtain 90% power with a 2.5% one-sided Type I error. A major issue with this type of study is that many experimental cancer therapies have toxic side-effects and, at the same time, do not provide benefit. For such drugs, it is desirable to minimize the number of patients exposed to the experimental regimen and further to minimize the duration of exposure for those who are exposed. Thus, it is highly desirable to do an early evaluation of data to stop the trial if no treatment benefit is emerging during the course of the trial. Such an evaluation must be carefully planned to 1) avoid an unplanned impact on the power of the study, and 2) to allow a realistic assessment of the emerging treatment effect.

1.7 A non-inferiority study for a new drug

The `nBinomial()` function presented above was specifically designed to work for noninferiority trial design as well as superiority designs. We consider a new treatment that is to be compared to a standard that has a successful treatment rate of 67.7%. An absolute margin of 7% is considered an acceptable noninferiority margin. The trial is to be powered at 90% with 2.5% Type I error (one-sided) using methods presented by Farrington and Manning [Farrington and Manning, 1990]. The function call `nBinomial(p1 = 0.677, p2 = 0.677, delta0 = 0.07)` shows that a fixed sample size of 1874 is adequate for this purpose. There are some concerns about these assumptions, however. First, the control group event rate may be incorrect. As the following code using event rates from 0.55 to 0.75 demonstrates, the required sample size may range from 1600 to over 2100.

```r
p <- c(0.55, 0.6, 0.65, 0.7, 0.75)
ceiling(nBinomial(p1 = p, p2 = p, delta0 = 0.07))
#> [1] 2117 2054 1948 1800 1611
```

More importantly, if the experimental group therapy does not work quite as well as control, there is a considerable dropoff in power to demonstrate non-inferiority. Thus, there may be value in planning an interim futility analysis to stop the trial if the success rate with experimental therapy is trending substantially worse than with control.

1.8 A diabetes outcomes trial example

Current regulatory standards for chronic therapies of diabetes require ensuring that a new drug in a treatment class does not have substantially inferior cardiovascular outcomes compared to an approved treatment or treatments [Center for Drug Evaluation and Research, 2008]. While we do not claim the designs for this example presented here would be acceptable to regulators, the specifics of the guidance provide a nice background for the use of the gsDesign package to derive group sequential designs that fit a given problem. The initial reason for presenting this example is that there is likely to be a genuine public health interest in showing any of the following for the two treatment arms compared:

- The two treatment arms are similar (equivalence).
- One arm is similar to or better than the other (non-inferiority).
- Either arm is superior to the other (2-sided testing of no difference).

The example is somewhat simplified here. We assume patients with diabetes have a risk of a cardiovascular event of about 1.5% per year and a 15%

dropout rate per year. If each arm has the same cardiovascular risk as the other, we would like to have 90% power to rule out a hazard ratio of 1.3 in either direction. Type I error if one arm has an elevated hazard ratio of 1.3 compared to the other should be 2.5% if one-sided. The trial is to enroll in 2 years and have a minimum follow-up of 4 years, leading to a total study time of 6 years. The sample size routine `nSurv()` is set up to handle this by making the null hypothesis a hazard ratio of 1.3 (`hr0 = 1.3` below) and the alternate hypothesis a hazard ratio of 1 (`hr = 1` below) to reflect equivalence. Our assumed rate for the both groups of $\lambda = \lambda = -\ln(1 - 0.015)$ under the alternate hypothesis is what we want to drive the sample size.

```
x <- nSurv(
  lambdaC = -log(1 - 0.015),
  hr0 = 1.3,
  hr = 1,
  eta = -log(0.85),
  alpha = 0.025,
  beta = 0.1,
  T = 6,
  minfup = 4
)
n <- ceiling(x$n / 2) * 2
d <- ceiling(x$d)
cat(paste("Sample size:", n, "Events:", d, "\n"))
#> Sample size: 12362 Events: 617
```

We note that the power for this sample size has been verified by simulation. This can be done with the simtrial package as follows; the numbers are not executed here. This verification uses the Schoenfeld approximation for the variance since the `simtrial::simfix()` function was not set up to save Cox model standard errors. One-thousand simulations estimated power at 90.7% when the planned minimum of 4 years of follow-up was obtained.

```
library(simtrial)
library(dplyr)

xx <- simfix(
  nsim = 1000,
  sampleSize = 12362,
  targetEvents = 617,
  totalDuration = 6,
  enrollRates = tibble::tibble(
    duration = 2,
    rate = 12362 / 2
  ),
  failRates = tibble::tibble(
    Stratum = "All",
    duration = 6,
    failRate = -log(1 - 0.015),
    hr = 1,
    dropoutRate = -log(0.85)
  ),
  timingType = 3
)
xx %>%
  mutate(se = sqrt(4 / Events)) %>%
  summarize(
    Events = mean(Events),
    Duration = mean(Duration),
    Power = mean(lnhr + qnorm(0.975) * se < log(1.3))
  )
#>   Events Duration Power
#>   617.53 6.000395 0.907
```

Generally, a confidence interval for the hazard ratio of experimental to control is used to express treatment differences at the end of this type of trial. A confidence interval will rule out the specified treatment differences consistently with testing if, for example, the same proportional hazards regression model is used for both the a Wald test and the corresponding confidence interval. The terminology of "control" and "experimental" is generally inappropriate when both therapies are approved. However, for this example it is generally the case that a new therapy is being compared to an established one and there may be some asymmetry when considering the direction of inference. Various questions arise concerning early stopping in a trial of this nature:

- While it would be desirable to stop early if the new therapy has a significantly lower cardiovascular event rate, a minimum amount of follow-up may be valuable to ensure longer-term safety and general acceptance of the results.
- If a trend emerges in favor of the experimental treatment, it will likely be possible to demonstrate non-inferiority prior to being able to demonstrate superiority. If the trial remains blinded until superiority is demonstrated or until the final planned analysis, full acceptance of a useful new therapy may be delayed. As noted above, the value of long-term safety data may be more important than an early stop based on "short-term" endpoint.
- From a sponsor's standpoint, it may be desirable to stop the trial if it becomes futile to demonstrate the experimental therapy is non-inferior to control; that is, there is an interim trend favoring control. However, if both treatment groups represent marketed products then from a public health standpoint it may be desirable to continue the trial to demonstrate a statistically significant advantage for the control treatment.

Chapter 2

Group sequential design theory and notation in brief

We begin by defining the distribution theory for the joint set of statistics used for testing in a group sequential design. While the primary purpose of the gsDesign package is to design group sequential trials, computing boundary crossing probabilities is the essential next step. Finally, we discuss the expected sample size for a group sequential design.

2.1 Distributional assumptions

We illustrate the distribution theory with a sequence of normal random variates. Let X_1, X_2,... be independent and identically distributed normal random variables with mean δ and variance σ^2. For some positive integer k, let $n_1 < n_2 ... < n_k$ represent fixed sample sizes where data will be analyzed and inference surrounding δ will be examined. This is referred to as a group sequential design. The first $k-1$ analyses are referred to as interim analyses, while the k^{th} analysis is referred to as the final analysis. For $i = 1, 2, ...k$ consider estimating δ with

$$\hat{\delta}_i = \sum_{j=1}^{n_i} X_j / n_i \tag{2.1}$$

The variance of $\hat{\delta}_i$ is σ^2/n_i and the corresponding statistical information for $\hat{\delta}_i$ is the reciprocal of the variance,

$$\mathcal{I}_i \equiv n_i/\sigma^2 \tag{2.2}$$

for $i = 1, 2, \ldots, k$. The value δ will be referred to as the natural parameter for this problem. We next define the standardized parameter $\theta = \delta/\sigma$ and its corresponding estimates (assuming σ known) as

$$\hat{\theta}_i = \sum_{j=1}^{n_i} X_j/(\sigma n_i) \tag{2.3}$$

which has variance $1/n_i$ for $i = 1, 2, \ldots, k$.

Next, for $i = 1, 2, \ldots, k$ we consider the test statistic

$$Z_i = \sqrt{n_i}\hat{\theta}_i = \sqrt{I_i}\hat{\delta}_i. \tag{2.4}$$

The test statistics Z_1, Z_2,...,Z_k follow a multivariate normal distribution with, for $1 \leq j \leq i \leq k$,

$$E\{Z_i\} = \sqrt{n_i}\delta/\sigma \equiv \sqrt{n_i}\theta \tag{2.5}$$

and

$$Cov(Z_j, Z_i) = \sqrt{n_j/n_i}, \tag{2.6}$$

or, equivalently,

$$E\{Z_i\} = \delta\sqrt{\mathcal{I}_i} \tag{2.7}$$

and

$$Cov(Z_j, Z_i) = \sqrt{\mathcal{I}_j/\mathcal{I}_i}. \tag{2.8}$$

Given the above formulations, we will tend to think in terms of the natural parameter δ and statistical information when considering estimation problems. When considering sample size, on the other hand, it is often convenient to think in terms of the standardized parameter θ.

Jennison and Turnbull [Jennison and Turnbull, 2000] refer to Equation 2.7 and Equation 2.8 as the canonical form and present several types of outcomes for which test statistics for comparing treatment groups take this form asymptotically. Specific examples in this manual consider 2-arm trials with binary or time-to-event outcomes. Note that when $\delta = 0$, the multivariate normal distribution expressed in Equation 2.7 and Equation 2.8 (or Equation 2.5 and Equation 2.6) depends only on $\mathcal{I}_i/\mathcal{I}_k = n_i/n_k$, $i = 1, 2, \ldots, k-1$. Scharfstein, Tsiatis and Robins [Scharfstein et al., 1997] provided a unifying theory to justify the above distribution theory when efficient test statistics are applied to test a single parameter in a parametric or semi-parametric model; they noted many cases where test statistics have been shown to be efficient and analyze and then simulate a study applying longitudinal data analysis. The application of the canonical distribution to a wide variety of problems is also discussed by Jennison and Turnbull [Jennison and Turnbull, 2000], among others. In this manual we consider binomial endpoint studies and time-to-event endpoint studies as our primary examples.

Computational methods for the gsDesign package related to the above multivariate normal distribution are described in Chapter 19 of Jennison and Turnbull [Jennison and Turnbull, 2000] and are not provided here. Note that other software programs such as EAST and the University of Wisconsin software use this distributional assumption for group sequential design computations.

2.2 Hypotheses and testing

We assume that the primary test the null hypothesis H_0: $\theta = 0$ against the alternative H_1: $\theta = \theta_1$ for a fixed standardized effect size $\theta_1 > 0$. Corresponding to θ_1 we have $\delta_1 = \theta_1\sigma$. We will work with θ most often. The values of θ and δ will be referred to as standardized and natural parameter treatment effects, respectively. We have arbitrarily assumed that $\theta, \delta > 0$ represents a treatment benefit and will refer to this case as a positive treatment effect. We assume further that there is interest in stopping early if there is good evidence to reject one hypothesis in favor of the other. For $i = 1, 2, \ldots, k-1$, interim cutoffs $l_i < u_i$ are set; final cutoffs $l_k \leq u_k$ are also set. For $i = 1, 2, \ldots, k$, the trial is stopped at analysis i to reject H_0 if $l_j < Z_j < u_j$, $j = 1, 2, \ldots, i-1$ and $Z_i \geq u_i$. If the trial continues until stage i, H_0 is not rejected at stage i, and $Z_i \leq l_i$ then H_1 is rejected in favor of H_0, $i = 1, 2, \ldots, k$. Thus, $3k$ parameters define a group sequential design: l_i, u_i, and $\mathcal{I}_i$, $i = 1, 2, \ldots, k$. Note that if $l_k < u_k$ there is the possibility of completing the trial without rejecting H_0 or H_1. We will often restrict $l_k = u_k$ so that one hypothesis is rejected.

2.3 Boundary crossing probabilities: `gsProbability()`

2.3.1 One-sided testing

We begin with a one-sided test. In this case there is no interest in stopping early for a lower bound and thus $l_i = -\infty$, $i = 1, 2, \ldots, k$. The probability of first crossing an upper bound at analysis i, $i = 1, 2, \ldots, k$, is

$$\alpha_i^+(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{Z_j < u_j\}\}$$

The Type I error is the probability of ever crossing the upper bound when $\theta = 0$. The value $\alpha_i^+(0)$ is commonly referred to as the amount of Type I error spent at analysis i, $1 \leq i \leq k$. The total upper boundary crossing probability for a trial is denoted in this one-sided scenario by

$$\alpha^+(\theta) \equiv \sum_{i=1}^{k} \alpha_i^+(\theta)$$

and the total Type I error by $\alpha^+(0)$. Assuming $\alpha^+(0) = \alpha$ the design will be said to provide a one-sided group sequential test at level α.

As an example, assume $k = 5$, $n_i = 100 \times i$, and $u_i = \Phi^{-1}(0.975) = 1.96$, $i = 1, 2, 3, 4, 5$. Thus, we are testing 5 times at a nominal 0.025 level at five equally spaced analyses. The function `gsProbability()` is a simple group sequential calculator to compute the probability of crossing bounds at each analysis as follows. `gsProbability()` requires a lower bound; we let $l_i = -20$, $1 \leq i \leq 5$ to effectively make the probability of crossing a lower bound 0.

```r
library(gsDesign)

k <- 5
x <- gsProbability(
  k = k,
  theta = 0,
  n.I = 1:k * 100,
  a = array(-20, k),
  b = array(qnorm(0.975), k)
)
tibble(
  Analysis = 1:x$k,
  N = x$n.I,
  "Upper bound" = x$upper$bound,
  "Nominal test level" = pnorm(
    x$upper$bound,
    lower.tail = FALSE
  ),
  "alpha+[i](0)" = x$upper$prob[, 1],
  "Cumulative alpha" = cumsum(x$upper$prob[, 1])
) |>
  gt() |>
  fmt_number(columns = 3:6, decimals = 3) |>
  tab_header(
    title = "Multiplicity problem with repeated testing"
  ) |>
  tab_options(data_row.padding = px(1))
```

Multiplicity problem with repeated testing

Analysis	N	Upper bound	Nominal test level	alpha+[i](0)	Cumulative alpha
1	100	1.960	0.025	0.025	0.025
2	200	1.960	0.025	0.017	0.042
3	300	1.960	0.025	0.012	0.054
4	400	1.960	0.025	0.009	0.063
5	500	1.960	0.025	0.008	0.071

Users of `gsProbability()` may want to look at the the list of items output since we have just provided a simple summary above. The help file for the routine provides details of what is returned, which can help you to produce simple summary tables such as the above. The table above shows that for the first two equally spaced analyses tested at the $\alpha = 0.025$ level that cumulative Type I error is already increased to 0.042. At 5 equally-spaced analyses, this increases to 0.071, nearly triple the nominal level used for testing at each analysis. These calculations are based on the multivariate normal distribution above.

In the above code, the call to `gsProbability()` returned a value into `x` which was then printed. The command `class(x)` following the above code will indicate that `x` has class `gsProbability`. The elements of this class are displayed as follows:

```
class(x)
#> [1] "gsProbability"
```

The names of components are:

```
names(x)
#> [1] "k"        "theta"    "n.I"      "lower"    "upper"    "en"
#> [7] "r"        "overrun"
```

Briefly, `k` is the number of analyses, `theta` a vector of standardized effect sizes, and `n.I` is a vector containing $\mathcal{I}_i$, $i = 1, 2, \ldots, k$. The value in r is a positive integer input to `gsProbability()` that is used to define how fine of a grid is used for numerical integration when computing boundary crossing probabilities; this is the same as input and will normally not be changed from the default of 18. The values of `en` and `overrun` will be discussed below in Section 2.4. This leaves the lists `lower` and `upper`, which have identical structure. We examine `upper` by

```
x$upper
#> $bound
#> [1] 1.959964 1.959964 1.959964 1.959964 1.959964
#>
#> $prob
#>             [,1]
#> [1,] 0.025000000
#> [2,] 0.016558911
#> [3,] 0.012070163
#> [4,] 0.009460567
#> [5,] 0.007769166
```

to see that it contains a vector `bound` which contains the values for u_i and upper boundary crossing probabilities in `prob[i,j]` for analysis i and the θ-value in `theta[j]` for `i=1,2,...,k` and `j` from 1 to the length of `theta`. Further documentation is in the online help file displayed using `help(gsProbability)`.

A Bonferroni adjustment maintains a Type I error of ≤ 0.025. For $1 \leq i \leq 5$ this would use the upper bound $u_i = \Phi^{-1}(1 - .025/5)$. Substituting `qnorm(1 - 0.025 / 5)` for `qnorm(0.975)` in the above call to `gsProbability()` yields an upper bound of 2.576 (nominal $p = 0.005$) and total Type I error of 0.016. Thus, the Bonferroni adjustment is overly conservative.

```
k <- 5
x <- gsProbability(
  k = k,
  theta = 0,
  n.I = 1:k * 100,
  a = array(-20, k),
  b = array(qnorm(1 - 0.025 / k), k)
)
tibble(
  Analysis = 1:x$k,
  N = x$n.I,
  "Upper bound" = x$upper$bound,
  "Nominal test level" = pnorm(
    x$upper$bound,
    lower.tail = FALSE
  ),
  "alpha+[i](0)" = x$upper$prob[, 1],
  "Cumulative alpha" = cumsum(x$upper$prob[, 1])
) |>
  gt() |>
  fmt_number(columns = 3:6, decimals = 3) |>
  tab_header(
    title = paste0(
      "Conservative control of Type I error ",
      "using a Bonferroni correction"
    )
  ) |>
  tab_options(data_row.padding = px(1))
```

Conservative control of Type I error using a Bonferroni correction

Analysis	N	Upper bound	Nominal test level	alpha+[i](0)	Cumulative alpha
1	100	2.576	0.005	0.005	0.005
2	200	2.576	0.005	0.004	0.009
3	300	2.576	0.005	0.003	0.012
4	400	2.576	0.005	0.002	0.014
5	500	2.576	0.005	0.002	0.016

The simple Bonferroni correction does not take advantage of the known correlations between tests and over corrects Type I error. This illustrates the rationale for finding bounds that control the total Type I error at a desired level such as $\alpha = 0.025$ without being overly conservative. We now illustrate bound calculation with the `gsDesign()` function to derive bounds for controlling Type I error under the multivariate normal distribution noted above. We do not fully explain now all of the code used to limit output to what is essential here. We note initially that we input the value `n.fix` as a sample size for a fixed design with no interim analyses. The resulting sample sizes at interims and final are inflated to retain power at the same level as such a fixed design.

```
k <- 5
x <- gsDesign(
  # Number of analyses
  k = k,
  # Superiority testing only,
  test.type = 1,
  # Equal Z-values for each bound
  sfu = "Pocock",
  # Equally spaced analysis timing
  timing = 1,
  # Assume fixed design would require N = k * 100
  n.fix = k * 100
)
x |>
  gsBoundSummary(
    exclude = c(
      "B-value", "Spending", "CP",
      "CP H1", "PP", "P(Cross) if delta=1",
      "~delta at bound"
    )
  ) |>
  gt() |>
  tab_header(
    title = "Pocock bounds for equally spaced analyses"
  ) |>
  tab_footnote("Now alpha = 0.025 is fully utilized.") |>
  tab_footnote(
    "Nominal p-value at bound",
    locations = cells_body(rows = 0:4 * 3 + 2, columns = 2)
  ) |>
  tab_footnote(
    paste0(
      "Null hypothesis probability of Type I error ",
      "at or before current analyses"
    ),
    cells_body(rows = 1:5 * 3, columns = 2)
  ) |>
  tab_options(data_row.padding = px(1))
```

We now see that although we cannot test at a nominal 0.025 level repeatedly, we can test at a nominal level of 0.0079 and control Type I error at $\alpha = 0.025$ without using the overly conservative Bonferroni nominal level of 0.0025.

2.3.2 Two-sided testing

With both lower and upper bounds for testing and any real value θ representing treatment effect we denote the probability of crossing the upper boundary at analysis i without previously crossing a bound by

$$\alpha_i(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\},$$

$i = 1, 2, \ldots, k$. The total probability of crossing an upper bound prior to crossing a lower bound is denoted by

Pocock bounds for equally spaced analyses

Analysis	Value	Efficacy
IA 1: 20%	Z	2.4132
N: 121	p (1-sided)[1]	0.0079
	P(Cross) if delta=0[2]	0.0079
IA 2: 40%	Z	2.4132
N: 242	p (1-sided)[1]	0.0079
	P(Cross) if delta=0[2]	0.0138
IA 3: 60%	Z	2.4132
N: 362	p (1-sided)[1]	0.0079
	P(Cross) if delta=0[2]	0.0183
IA 4: 80%	Z	2.4132
N: 483	p (1-sided)[1]	0.0079
	P(Cross) if delta=0[2]	0.0219
Final	Z	2.4132
N: 604	p (1-sided)[1]	0.0079
	P(Cross) if delta=0[2]	0.0250

Now alpha = 0.025 is fully utilized.
[1]Nominal p-value at bound
[2]Null hypothesis probability of Type I error at or before current analyses

$$\alpha(\theta) \equiv \sum_{i=1}^{k} \alpha_i(\theta).$$

Next, we consider analogous notation for the lower bound. For $i = 1, 2, \ldots, k$ denote the probability of crossing a lower bound at analysis i without previously crossing any bound by

$$\beta_i(\theta) = P_\theta\{\{Z_i \le l_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}.$$

For symmetric testing for analysis i we would have $l_i = -u_i$, $\beta_i(0) = \alpha_i(0)$, $i = 1, 2, \ldots, k$. The total lower boundary crossing probability in this case is written as $\beta(\theta) = \sum_{i=1}^{k} \beta_i(\theta)$. The total lower boundary crossing probability for a trial is denoted by

$$\beta(\theta) \equiv \sum_{i=1}^{k} \beta_i(\theta).$$

To extend the one-sided example using repeated testing at a 0.025 level to two-sided testing at the 0.05 level, try the commands

```
b <- array(qnorm(0.975), 3)
x2 <- gsProbability(
  k = 3,
  theta = 0,
  n.I = c(100, 200, 300),
  a = -b,
  b = b
)
x2
#>                  Lower bounds   Upper bounds
#>   Analysis  N    Z   Nominal p  Z   Nominal p
#>          1 100 -1.96     0.025 1.96     0.025
#>          2 200 -1.96     0.025 1.96     0.025
#>          3 300 -1.96     0.025 1.96     0.025
#>
#> Boundary crossing probabilities and expected sample size assume
#> any cross stops the trial
#>
#> Upper boundary
#>          Analysis
#>   Theta     1      2      3  Total  E{N}
#>       0 0.025 0.0166 0.0121 0.0536 286.7
#>
#> Lower boundary
#>          Analysis
#>   Theta     1      2      3  Total
#>       0 0.025 0.0166 0.0121 0.0536
```

The fact that a lower bound can be crossed before crossing an upper bound means that after the first interim analysis the upper boundary crossing probability here should be less than it was for the one-sided computation performed previously. To examine this further, we print the upper boundary crossing probability at each analysis for the above one-sided and two-sided examples, respectively, to see that there is a small difference:

```
x$upper$prob
#>              [,1]       [,2]
#> [1,] 0.007906997 0.20587421
#> [2,] 0.005855842 0.26025148
#> [3,] 0.004509295 0.20860438
#> [4,] 0.003655118 0.14019500
#> [5,] 0.003072747 0.08507493
```

Group sequential designs most often employ more stringent bounds at early interim analyses than later. We modify the above example to demonstrate this.

```
b <- qnorm(0.975) / sqrt(c(1, 2, 3) / 3)
b
#> [1] 3.394757 2.400456 1.959964
```

```
x2b <- gsProbability(
  k = 3,
  theta = 0,
  n.I = c(100, 200, 300),
  a = -b,
  b = b
)
x2b
#>                Lower bounds   Upper bounds
#>   Analysis  N    Z   Nominal p  Z   Nominal p
#>          1 100 -3.39    0.0003 3.39    0.0003
#>          2 200 -2.40    0.0082 2.40    0.0082
#>          3 300 -1.96    0.0250 1.96    0.0250
#>
#> Boundary crossing probabilities and expected sample size assume
#> any cross stops the trial
#>
#> Upper boundary
#>           Analysis
#>   Theta     1     2      3  Total  E{N}
#>       0 3e-04 0.008 0.0195 0.0279 298.3
#>
#> Lower boundary
#>           Analysis
#>   Theta     1     2      3  Total
#>       0 3e-04 0.008 0.0195 0.0279
```

By setting the interim boundaries to be substantially higher than $\Phi^{-1}(0.975) = 1.96$ we have drastically reduced the excess Type I error caused by multiple testing while still testing at the nominal 0.05 (2-sided) level at the final analysis. Thus, minimal adjustment to the final boundary should be required when employing such a strategy. Precise control of Type I error when using either equal bounds or adjusting relative sizes of bounds using the square root of sample size is discussed further in Section 6.2.

2.4 Expected sample size

We denote the sample size at analysis i by n_i, $i = 1, 2, \ldots, k$ and the sample size at the time a boundary is crossed by N. The expected sample size is

$$E_\theta\{N\} = \sum_{i=1}^{k} n_i \times (\alpha_i(\theta) + \beta_i(\theta)).$$

Values of $E_\theta\{N\}$ corresponding to θ-values input to `gsProbability()` in `theta` are output in the vector `en` returned by that function. In the one- and two-sided examples above we only had a single element 0 in `theta` and the expected sample sizes rounded to 293 and 298, respectively; these were labeled `E{N}` in the printed output. Since the probability of crossing a boundary at an interim analysis was small, the trial usually proceeds to the final analysis with 300 observations. We consider an additional θ-value to demonstrate that the average sample number can be substantially smaller than the maximum sample size:

```
x2c <- gsProbability(
  k = 3,
  theta = c(0, 0.3),
  n.I = c(100, 200, 300),
  a = -b,
  b = b
)
x2c
#>                  Lower bounds   Upper bounds
#>   Analysis   N    Z   Nominal p  Z   Nominal p
#>          1 100 -3.39    0.0003 3.39    0.0003
#>          2 200 -2.40    0.0082 2.40    0.0082
#>          3 300 -1.96    0.0250 1.96    0.0250
#>
#> Boundary crossing probabilities and expected sample size assume
#> any cross stops the trial
#>
#> Upper boundary
#>           Analysis
#>   Theta      1      2      3  Total   E{N}
#>     0.0 0.0003 0.0080 0.0195 0.0279 298.3
#>     0.3 0.3465 0.6209 0.0320 0.9994 168.6
#>
#> Lower boundary
#>           Analysis
#>   Theta     1     2      3  Total
#>     0.0 3e-04 0.008 0.0195 0.0279
#>     0.3 0e+00 0.000 0.0000 0.0000
```

Thus, assuming a positive treatment effect, the average sample number was 169 compared to 298 when there was no treatment difference.

Chapter 3

Continuous and integer sample size

The gsDesign package has historically used continuous values for sample size and event counts at the time of design. This has the advantage that timing of analyses can be precise in terms of specified timing as specified by fraction of final sample size (normal and binary endpoints) or fraction of targeted events (time-to-event outcomes). Disadvantages include ambiguity when updating designs at time of analysis based on integer sample size or event counts as will be demonstrated below. We illustrate basic implementation of integer-based sample size in this chapter and provide further examples throughout the book. The `toInteger()` function will convert designs to integer-based sample size. For designs for time-to-event endpoints created using the `gsSurv()` or `gsSurvCalendar()` functions, integer-based event counts are also produced by the `toInteger()` design conversion.

We consider a simple design with a user-defined endpoint with a fixed design sample size of `n.fix = 1000` with default 1-sided Type I error $\alpha = 0.025$ and 90% power.
We add a single interim analysis after 60% of the trial is available for analysis using only a superiority bound for the interim analysis (`test.type = 1`) and apply a Hwang et al. [1990] spending function (`sfu = sfHSD`) with spending parameter $\gamma = -3$ (`sfupar = -3`). We see immediately that the derived design does not have integer sample sizes at analyses.

```r
library(gsDesign)

x <- gsDesign(
  n.fix = 1000,
  k = 2,
  timing = 0.6,
  test.type = 1,
  sfu = sfHSD,
  sfupar = -3
)
x$n.I
#> [1]  611.3689 1018.9482
```

The integer-based sample size for this is obtained as follows:

```
y <- toInteger(x)
y$n.I
#> [1]  611 1019
```

The interim sample size above has simply been rounded while the final sample size has been rounded up. The `ratio` parameter in the `toInteger()` function controls how the full trial sample size rounding is done. The value `ratio = 1` assumes 1:1 randomization and, thus, an even total sample size.

```
y <- toInteger(x, ratio = 1)
y$n.I
#> [1]  611 1020
```

In the `toInteger()` function, `ratio` is used to specify a conversion of the input design to round up to the next even multiple of `ratio + 1` for the total sample size. Thus, if the randomization ratio were 5:2, we might want the sample size to be an even multiple of 7 and would would specify:

```
toInteger(x, ratio = 6)$n.I
#> [1]  611 1022
```

There is also a parameter `roundUpFinal` with a default value of `TRUE`. If `FALSE`, rather than rounding the final value up, it is just rounded. In our example above, this makes no difference.

```
toInteger(x, ratio = 2, roundUpFinal = FALSE)$n.I
#> [1]  611 1020
```

For the design with 1:1 randomization and an even sample size, rather than the input information fraction of 0.6 at analysis 1 we have slightly smaller value

```
y$n.I[1] / y$n.I[2]
#> [1] 0.5990196
```

Also, rather than the originally targeted power of 90%, we have a total power of

```
100 * sum(y$upper$prob[, 2])
#> [1] 90.0303
```

Now we update the design assuming instead of 2 analyses after 611 and 1020 observations we have 3 analyses as shown in the code below. Most of the code in the `gsDesign()` call is copying in parameters from the design defined in the integer-based sample size design `y` above. We note the Z-value bounds for efficacy under the asymptotic distributional assumptions of the previous chapter.

```r
yu <- gsDesign(
  # Assume 3 analyses actually done
  k = 3, n.I = c(400, 700, 1100),
  # Remaining parameters copied from original design
  maxn.IPlan = y$n.I[y$k],
  test.type = y$test.type,
  alpha = y$alpha, beta = y$beta, astar = y$astar,
  sfu = y$upper$sf, sfupar = y$upper$param,
  sfl = y$lower$sf, sflpar = y$lower$param,
  delta = y$delta, delta1 = y$delta1, delta0 = y$delta0,
)
yu$upper$bound
#> [1] 2.754625 2.444650 2.038284
```

A key parameter in the above that leads to some ambiguity in the case continuous sample size is `maxn.IPlan`, the planned sample size or, for time-to-event outcomes, the planned final analysis event count. The prescribed way to do this is as follows which is identical to the coding approach above for `yu`, replacing the integer sample size design in `y` with the continuous sample size design in `x`:

```r
xu <- gsDesign(
  # Assume 3 analyses actually done
  k = 3, n.I = c(400, 700, 1100),
  # Remaining parameters copied from original design
  maxn.IPlan = x$n.I[x$k],
  test.type = y$test.type,
  alpha = x$alpha, beta = x$beta, astar = x$astar,
  sfu = x$upper$sf, sfupar = x$upper$param,
  sfl = x$lower$sf, sflpar = x$lower$param,
  delta = x$delta, delta1 = x$delta1, delta0 = x$delta0,
)
xu$upper$bound
#> [1] 2.754051 2.443665 2.038577
```

This gives a slightly different result than if we specify the rounded (integer) sample size from the original design in `maxn.IPlan`, the only change from the code above.

```r
xu <- gsDesign(
  # Assume 3 analyses actually done
  k = 3, n.I = c(400, 700, 1100),
  # Remaining parameters copied from original design
  maxn.IPlan = 1020,
  test.type = y$test.type,
  alpha = x$alpha, beta = x$beta, astar = x$astar,
  sfu = x$upper$sf, sfupar = x$upper$param,
  sfl = x$lower$sf, sflpar = x$lower$param,
  delta = x$delta, delta1 = x$delta1, delta0 = x$delta0,
)
xu$upper$bound
#> [1] 2.754625 2.444650 2.038284
```

With the integer-based design in `y`, `maxn.IPlan` will be 1020 if gotten from `y$n.I[y$k]` or entered directly from a summary table removing any ambiguity about how bounds should be updated at the time of study analyses when event counts or sample size realized will generally be different from the original plan simply due to logistical considerations.

Chapter 4

Applying the default group sequential design

4.1 Default parameters

We are now prepared to demonstrate derivation of group sequential designs using default parameters with the `gsDesign()` function. Along with this, we discuss the `gsDesign` class returned by `gsDesign()` and its associated standard print and plot functions. We then apply this default group sequential design to each of our motivational examples. The main parameters in `gsDesign()` will be explained in more detail in Chapter 6 through Chapter 8.

The main parameter defaults that you need to know about are as follows:

1. Overall Type I error (α, one-sided): `alpha = 0.025`.
2. Overall Type II error ($\beta = 1 - \text{power}$): `beta = 0.1`.
3. Two interim analyses equally spaced at 1/3 and 2/3 of the way through the trial plus the final analysis: `k=3`.
4. `test.type = 4`, which specifies all of the following:
   - Asymmetric boundaries, which means we may stop the trial for futility or superiority at an interim analysis.
   - β-spending is used to set the lower stopping boundary. This means that the spending function controls the incremental amount of Type II error at each analysis, $\beta_i(\theta_1)$, $i = 1, 2, \ldots, K$.
   - Non-binding lower bound. Lower bounds are sometimes considered as guidelines, which may be ignored during the course of the trial. Since Type I error is inflated if this if futility bounds are ignored, regulators often demand that the lower bounds be ignored when computing Type I error. That is, Type I error is computed using $\alpha^+(\theta)$ rather than $\alpha(\theta)$.
5. Hwang-Shih-DeCani spending functions for the upper bound (`sfu = sfHSD`) with γ-parameter `sfupar = -4` and lower bound (`sfl = sfHSD`) with γ-parameter `sflpar = -2`. This provides a conservative, O'Brien-

Fleming-like superiority bound and a less conservative lower bound. Spending functions will be discussed in detail in Chapter 8.

6. The following parameters are related to numerical accuracy and will not be discussed further here as they generally would not be changed by the user: `tol = 0.000001`, `r = 18`. Further information is in the help file.
7. The input variable `endpoint` (default is `NULL`) at present impacts default options for plots approximating the treatment effect at a boundary. If `endpoint = "binomial"` then the y-axis will change to a default label $\widehat{p}_C - \widehat{p}_E$; for a study with a time-to-event outcome created with `gsSurv()` this is not needed.
8. `delta1` (default `1`) indicates the alternative hypothesis value on the natural (δ) parameter scale; e.g., log(HR) or risk difference. This is used to scale the treatment effect plot.
9. `delta0` is the null hypothesis value on the natural (δ) parameter scale. Generally, this will be `0`, but may be changed if you are testing for noninferiority or, as in a vaccine study, supersuperiority.
10. `nFixSurv` (default of `0`) is used to indicate the sample size for a fixed design for a survival trial. This is not needed and not computed if `gsSurv()` is used to derive a design for the time-to-event endpoint, so it is not likely to be used. If used, `n.fix` would indicate the number of endpoints for this trial to be powered as specified. By providing `nFixSurv`, printed output from a `gsDesign` object will include the total sample size as well as the number of events at interim and final analysis.
11. The following parameters are used to reset bounds when timing of analyses are changed from the original design and will be discussed in Section 8.4:
    - `maxn.IPlan`, if resetting timing of analyses, this contains the statistical information/sample size/number of events at the originally planned final analysis.
    - `n.I`, if `maxn.IPlan > 0` this is a vector of length `k` containing actual statistical information/sample size/number of events at each analysis.

4.2 Sample size ratio for a group sequential design compared to a fixed design.

In Chapter 2 and its subsections we gave distributional assumptions, defined testing procedures and denoted probabilities for boundary crossing. Consider a trial with a fixed design (no interim analyses) with power 100(1−β) and level α (1-sided). Denote the sample size as N_{fix} and statistical information for this design as $\mathcal{I}_{fix}$. For a group sequential design as noted above, we denote the information ratio (inflation factor) comparing the information planned for the final analysis of a group sequential design compared to a fixed design as

$$r = \mathcal{I}_k/\mathcal{I}_{fix} = n_k/N_{fix}. \tag{4.1}$$

This ratio is independent of the θ- or δ-value for which the trial is powered as long as the information (sample size) available at each analysis increases proportionately with $\mathcal{I}_{fix}$ and the boundaries for the group sequential design remain unchanged; see, for example, Jennison and Turnbull [Jennison and Turnbull, 2000]. Because of this, the default for `gsDesign()` is to print the sample size ratios $r_i = \mathcal{I}_i/\mathcal{I}_k$, $i = 1, 2, \ldots, k$ when the default value of `n.fix = 1` is used. With larger values of `n.fix`, a column labeled `N` is provided to give the sample size or number of events at each analysis. We demonstrate in the following subsections how to set `n.fix` to apply to our motivating examples.

4.3 The default call to `gsDesign()`

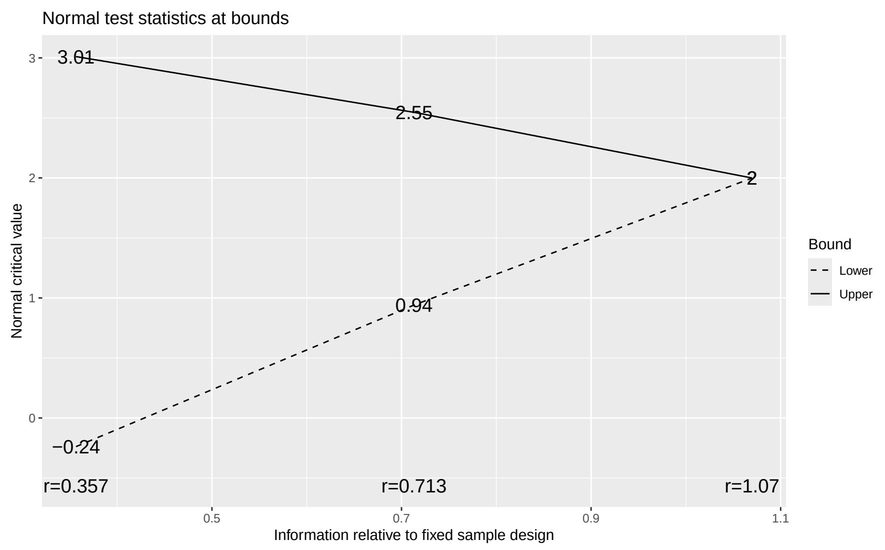
We begin with the call `x <- gsDesign()` to generate a design using all default arguments. The next line prints a summary of `x`; this produces the same effect as `print(x)` or `print.gsDesign(x)`. Note that while the total Type I error is 0.025, this assumes the lower bound is ignored if it is crossed; looking lower in the output we see the total probability of crossing the upper boundary at any analysis when the lower bound stops the trial is 0.0233. Had the option `x <- gsDesign(test.type = 3)` been run, both of these numbers would assume the trial stops if the lower bound stopped and thus would both be 0.025.

```
library(gsDesign)

x <- gsDesign()
x
#> Asymmetric two-sided group sequential design with
#> 90 % power and 2.5 % Type I Error.
#> Upper bound spending computations assume
#> trial continues if lower bound is crossed.
#>
#>            Sample
#>             Size    ----Lower bounds----  ----Upper bounds-----
#>   Analysis Ratio*   Z   Nominal p Spend+  Z   Nominal p Spend++
#>          1  0.357 -0.24    0.4057 0.0148 3.01    0.0013  0.0013
#>          2  0.713  0.94    0.8267 0.0289 2.55    0.0054  0.0049
#>          3  1.070  2.00    0.9772 0.0563 2.00    0.0228  0.0188
#>      Total                        0.1000                 0.0250
#> + lower bound beta spending (under H1):
#>  Hwang-Shih-DeCani spending function with gamma = -2.
#> ++ alpha spending:
#>  Hwang-Shih-DeCani spending function with gamma = -4.
#> * Sample size ratio compared to fixed design with no interim
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>    Theta      1      2      3  Total   E{N}
#>   0.0000 0.0013 0.0049 0.0171 0.0233 0.6249
#>   3.2415 0.1412 0.4403 0.3185 0.9000 0.7913
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>    Theta      1      2      3  Total
#>   0.0000 0.4057 0.4290 0.1420 0.9767
#>   3.2415 0.0148 0.0289 0.0563 0.1000
```

A plot of study bounds on the Z-value scale is provided by:

```
p <- plot(x)
p
```



There are several other plots available as will be discussed below. A brief textual summary of the design is obtained with:

```
summary(x)
```

```
#> Asymmetric two-sided group sequential design with
#> non-binding futility bound, 3 analyses, sample size 2, 90
#> percent power, 2.5 percent (1-sided) Type I error. Efficacy
#> bounds derived using a Hwang-Shih-DeCani spending function
#> with gamma = -4. Futility bounds derived using a
#> Hwang-Shih-DeCani spending function with gamma = -2.
```

A tabular summary of bounds is generated with

```
gsBoundSummary(x)
#>                Analysis              Value Efficacy Futility
#>               IA 1: 33%                  Z   3.0107  -0.2387
#>  N/Fixed design N: 0.36        p (1-sided)   0.0013   0.5943
#>                            ~delta at bound   1.5553  -0.1233
#>                        P(Cross) if delta=0   0.0013   0.4057
#>                        P(Cross) if delta=1   0.1412   0.0148
#>               IA 2: 67%                  Z   2.5465   0.9411
#>  N/Fixed design N: 0.71        p (1-sided)   0.0054   0.1733
#>                            ~delta at bound   0.9302   0.3438
#>                        P(Cross) if delta=0   0.0062   0.8347
#>                        P(Cross) if delta=1   0.5815   0.0437
#>                   Final                  Z   1.9992   1.9992
#>  N/Fixed design N: 1.07        p (1-sided)   0.0228   0.0228
#>                            ~delta at bound   0.5963   0.5963
#>                        P(Cross) if delta=0   0.0233   0.9767
#>                        P(Cross) if delta=1   0.9000   0.1000
```

Above we have seen standard output for `gsDesign()`. To access individual items of information about what is returned from the above, use `names(x)` to list the elements of `x`. Type `help(gsDesign)` to get full documentation of the class `gsDesign` returned by the `gsDesign()` function; the documentation website at https://keaven.github.io/gsDesign/ can also be useful. To view an individual element of `x` type, for example, `x$delta`, the standardized effect size for the design (θ from).

Other elements of `x` can be accessed in the same way, and we will use these to display aspects of designs in further examples. Of particular interest are the elements `upper` and `lower`. These are both objects containing multiple variables concerning the upper and lower boundaries and boundary crossing probabilities. Type `names(x$upper)` to show what these variables are. The upper boundary can be shown with the command `x$upper$bound`. As an example plot, enter `plot(x, plottype=2))` for a power plot. The argument `plottype` can run from 1 (the default) to 8. The options not already noted plot approximate effect sizes at boundaries (`plottype=3`; see `plottype=8` for hazard ratio), conditional power at boundaries (`plottype=4`), α- and β-

spending functions (`plottype=5`), expected sample size by underlying treatment difference (`plottype=6`), B-values at boundaries (`plottype=7`), and approximate hazard ratio at boundaries for designs for time-to-event outcomes (`plottype=8`).

4.4 Applying the default design to the CAPTURE example

The sample size ratios for each analysis relative to a fixed design sample size from Equation 4.1 can be obtained as follows:

```
x$n.I
#> [1] 0.3566277 0.7132555 1.0698832
```

These will be applied to each of our examples. Recall from the CAPTURE trial that we had a binomial outcome and wished to detect a reduction in the primary endpoint from a 15% event rate in the control group to a 10% rate in the experimental group. While we consider 80% power elsewhere, we stick with the default of 90% here. A group sequential design with 90% power and 2.5% Type I error has the same bounds as shown previously. The sample size at each analysis is obtained as follows (continuing the code just above):

```
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1)
n.fix
#> [1] 1834.641
```

```
n.fix * x$n.I
#> [1]  654.284 1308.568 1962.852
```

Rounding up to an even number for the final analysis, we see from the above that a while a fixed design requires 1836 patients, employing the default group sequential design inflates the sample size requirement to 1964. Interim analyses would be performed after approximately 655 and 1309 patients.

The group sequential design can be derived directly by replacing the input parameter `n.fix` with the sample size from a fixed design trial as follows:

```
library(dplyr)

n.I <- nBinomial(p1 = 0.15, p2 = 0.1)
# delta1 describes natural parameter risk difference
x <- gsDesign(
  n.fix = n.I,
  delta1 = 0.05,
  endpoint = "binomial"
) |> toInteger(ratio = 1)
```

```
x$n.I
#> [1]  654 1309 1964
```

The `toInteger()` function converts the continuous sample size created by `gsDesign()` to an integer-based sample size design. The argument `ratio = 1` in the `toInteger()` call indicates 1:1 randomization (experimental:control) and will round up the final sample size to an even integer. We will also see below that this yields slightly above the targeted 90% power. Printing the design now replaces the sample size ratio with the actual sample sizes at each analysis.

```
x
#> Asymmetric two-sided group sequential design with
#> 90 % power and 2.5 % Type I Error.
#> Upper bound spending computations assume
#> trial continues if lower bound is crossed.
#>
#>                   ----Lower bounds----  ----Upper bounds-----
#>   Analysis   N    Z   Nominal p Spend+  Z   Nominal p Spend++
#>          1  654 -0.24    0.4053 0.0148 3.01    0.0013  0.0013
#>          2 1309  0.94    0.8267 0.0289 2.55    0.0054  0.0049
#>          3 1964  2.00    0.9772 0.0563 2.00    0.0228  0.0188
#>      Total                      0.1000                 0.0250
#> + lower bound beta spending (under H1):
#>  Hwang-Shih-DeCani spending function with gamma = -2.
#> ++ alpha spending:
#>  Hwang-Shih-DeCani spending function with gamma = -4.
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>    Theta      1      2      3  Total   E{N}
#>   0.0000 0.0013 0.0049 0.0171 0.0233 1146.9
#>   0.0757 0.1410 0.4406 0.3186 0.9001 1452.4
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>    Theta      1      2      3  Total
#>   0.0000 0.4053 0.4294 0.1420 0.9767
#>   0.0757 0.0148 0.0289 0.0562 0.0999
```

Rather than printing the design as above, we recommend using the `summary()` function for a textual summary and `gsBoundSummary()` for a table summarizing bounds.

```
summary(x)
```

```
#> Asymmetric two-sided group sequential design with
#> non-binding futility bound, 3 analyses, sample size 1964,
#> 90 percent power, 2.5 percent (1-sided) Type I error.
#> Efficacy bounds derived using a Hwang-Shih-DeCani spending
#> function with gamma = -4. Futility bounds derived using a
#> Hwang-Shih-DeCani spending function with gamma = -2.
```

```
x |> gsBoundSummary(deltaname = "Risk difference")
#>   Analysis                            Value Efficacy Futility
#>  IA 1: 33%                                Z   3.0113  -0.2397
#>     N: 654                      p (1-sided)   0.0013   0.5947
#>              ~Risk difference at bound   0.0778  -0.0062
#>                P(Cross) if Risk difference=0   0.0013   0.4053
#>             P(Cross) if Risk difference=0.05   0.1410   0.0148
#>  IA 2: 67%                                Z   2.5468   0.9413
#>    N: 1309                      p (1-sided)   0.0054   0.1733
#>              ~Risk difference at bound   0.0465   0.0172
#>                P(Cross) if Risk difference=0   0.0062   0.8347
#>             P(Cross) if Risk difference=0.05   0.5815   0.0437
#>      Final                                Z   1.9992   1.9992
#>    N: 1964                      p (1-sided)   0.0228   0.0228
#>              ~Risk difference at bound   0.0298   0.0298
#>                P(Cross) if Risk difference=0   0.0233   0.9767
#>             P(Cross) if Risk difference=0.05   0.9001   0.0999
```

The risk difference at bound line gives an approximate minimum difference in event rates that will result in crossing a bound. This should work well when the average risk across arms is the same as the design average, in this case `(0.15 + 0.1) / 2 = 0.125`. For instance, looking at an observed risk difference of approximately 0.0078 at interim 1 when the average risk of the 2 arms is approximately 0.125 yields a Z-statistic close to that in the above table (3.01)

```
# Risk reduction uses x1, n1 for control,
# x2, n2 for experimental group.
testBinomial(
  x1 = round((0.125 + 0.0778 / 2) * 327), # 54 events
  x2 = round((0.125 - 0.0778 / 2) * 327), # 28 events
  n1 = 327, n2 = 327
)
#> [1] 3.070134
```

However, if the overall absolute risk is different this approximation will not work well:

```r
# Lower risk
testBinomial(
  x1 = round((0.1 + 0.0778 / 2) * 327), # 45 events
  x2 = round((0.1 - 0.0778 / 2) * 327), # 20 events
  n1 = 327, n2 = 327
)
#> [1] 3.267492
```

```r
# Higher risk
testBinomial(
  x1 = round((0.15 + 0.0778 / 2) * 327), # 62 events
  x2 = round((0.15 - 0.0778 / 2) * 327), # 36 events
  n1 = 327, n2 = 327
)
#> [1] 2.848471
```

4.4.1 Simulation of a binomial design

Testing at each analysis can be performed using the Miettinen and Nurminen [Miettinen and Nurminen, 1985] method. Simulation to verify the normal approximation is adequate for comparing binomial event rates can be performed using the functions `simBinomial` and `testBinomial` using the CAPTURE fixed design from above with `N = 1836`. This is only set up for fixed design, but suggests that the normal approximations used for power and Type I error calculations are quite good.

```r
# Show that power is 90% using simulation
Z <- simBinomial(
  nsim = 100000,
  p1 = 0.15,
  p2 = 0.1,
  n1 = 1836 / 2,
  n2 = 1836 / 2
)
mean(Z >= qnorm(0.975))
#> [1] 0.90273
```

Under the null hypothesis, we have:

```r
# Show that Type I error is 2.5%
Z <- simBinomial(
  nsim = 100000,
  p1 = 0.125,
  p2 = 0.125,
  n1 = 1836 / 2,
  n2 = 1836 / 2
)
mean(Z >= qnorm(0.975))
#> [1] 0.0252
```

4.4.2 Applying the default design to the noninferiority example

The fixed noninferiority design for a binomial comparison is the same as above, only changing the `nBinomial()` call to

```
n.fix <- nBinomial(p1 = 0.677, p2 = 0.677, delta0 = -0.07)
```

```
# delta0 is minus the non-inferiority margin,
# in this case, we are trying to rule out
# a risk increase over control of 0.07 or more.
ni_design <- gsDesign(
  n.fix = n.fix,
  delta0 = -0.07,
  delta1 = 0,
  endpoint = "binomial"
) |> toInteger(ratio = 1)
ni_design |> gsBoundSummary(deltaname = "Risk difference")
#>   Analysis                              Value Efficacy Futility
#>  IA 1: 33%                                  Z   3.0107  -0.2386
#>     N: 668                        p (1-sided)   0.0013   0.5943
#>                       ~Risk difference at bound   0.0389  -0.0786
#>             P(Cross) if Risk difference=-0.07   0.0013   0.4057
#>                 P(Cross) if Risk difference=0   0.1412   0.0148
#>  IA 2: 67%                                  Z   2.5465   0.9412
#>    N: 1336                        p (1-sided)   0.0054   0.1733
#>                       ~Risk difference at bound  -0.0049  -0.0459
#>             P(Cross) if Risk difference=-0.07   0.0062   0.8347
#>                 P(Cross) if Risk difference=0   0.5815   0.0437
#>      Final                                  Z   1.9992   1.9992
#>    N: 2004                        p (1-sided)   0.0228   0.0228
#>                       ~Risk difference at bound  -0.0283  -0.0283
#>             P(Cross) if Risk difference=-0.07   0.0233   0.9767
#>                 P(Cross) if Risk difference=0   0.9000   0.1000
```

We examine the risk difference approximations at the efficacy and futility bounds at analysis 2. First, for the futility bound 468 events in the experimental group vs. 437 in the control is a big enough increase in risk to declare futility for an eventual non-inferiority finding with an allowable increase in risk of no more than 0.07:

```r
testBinomial(
  delta0 = -0.07, n1 = 668, n2 = 668,
  x1 = round((0.677 - 0.0459 / 2) * 668), # 437 events
  x2 = round((0.677 + 0.0459 / 2) * 668) # 468 events
)
#> [1] 0.9244426
```

For the efficacy bound 454 experimental events in the experimental group vs. 451 in the control group is sufficient at the second interim to declare non-inferiority (i.e., rule out more than a 0.07 risk difference in the experimental group vs. control):

```r
testBinomial(
  delta0 = -0.07, n1 = 668, n2 = 668,
  x1 = round((0.677 - 0.0049 / 2) * 668), # 451 events
  x2 = round((0.677 + 0.0049 / 2) * 668) # 454 events
)
#> [1] 2.564413
```

We see these approximations are quite reasonable when to overall average event rate is 0.677.

4.5 Applying the default design to the cancer trial example

For trials with time-to-event outcomes, the variable `n.fix` in `gsDesign()` needed is the number of events from a fixed design trial. The reader may wish to refer to Jennison and Turnbull [Jennison and Turnbull, 2000] for further background; we also discuss distributional assumptions further in Section 7.3.2. We begin with the code from the fixed design trial for the cancer trial example from Section 1.6. Next, we call to `gsDesign()` with `n.fix` equal to the number of events for a fixed trial design. The value `ssratio`, the sample size ratio at each analysis compared to the fixed design sample size is then shown. Note that the values are the same as shown in the first output of this example above. The inflation in the total sample size is the same as for the number of events required if enrollment duration and dropout rates are not changed; that is, the sample size required for a group sequential design with the default interim analysis plan is inflated by multiplying by 1.07. In the last line of code below, we demonstrate a plot showing the approximate hazard ratios required to cross a bound.

```r
x <- nSurv(
  lambdaC = log(2) / 6,
  hr = 0.7,
  eta = -log(0.95) / 12,
  minfup = 6,
  T = 36
)
y <- gsDesign(n.fix = x$d)
y$n.I
#> [1] 117.3648 234.7296 352.0945
```

```r
y <- gsSurv(
  lambdaC = log(2) / 6,
  hr = 0.7,
  eta = -log(0.95) / 12,
  minfup = 6,
  T = 36
) |> toInteger()
y$n.I
```

The overall design summary is:

```r
summary(y)
```

```
#> Asymmetric two-sided group sequential design with
#> non-binding futility bound, 3 analyses, sample size 353, 90
#> percent power, 2.5 percent (1-sided) Type I error. Efficacy
#> bounds derived using a Hwang-Shih-DeCani spending function
#> with gamma = -4. Futility bounds derived using a
#> Hwang-Shih-DeCani spending function with gamma = -2.
```

The boundary summary table is obtained with:

```r
y |> gsBoundSummary(deltaname = "HR")
#>   Analysis            Value Efficacy Futility
#>  IA 1: 33%                Z   3.0107  -0.2387
#>     N: 118      p (1-sided)   0.0013   0.5943
#>                ~HR at bound   0.5736   1.0451
#>            P(Cross) if HR=0   0.0013   0.4057
#>            P(Cross) if HR=1   0.1412   0.0148
#>  IA 2: 67%                Z   2.5465   0.9411
#>     N: 235      p (1-sided)   0.0054   0.1733
#>                ~HR at bound   0.7172   0.8844
#>            P(Cross) if HR=0   0.0062   0.8347
#>            P(Cross) if HR=1   0.5815   0.0437
#>      Final                Z   1.9992   1.9992
#>     N: 353      p (1-sided)   0.0228   0.0228
#>                ~HR at bound   0.8081   0.8081
#>            P(Cross) if HR=0   0.0233   0.9767
#>            P(Cross) if HR=1   0.9000   0.1000
```

In order to update the design, we consider the planned maximum number of events and the achieved events at each analysis. For example, if we have 125, 250 and 365 events at the 3 analyses we update bounds as below. Note that if this is a mixture of planned (later analyses) and actual events (earlier analyses), this is fine. Is also possible to have fewer or more analyses than in the original design.

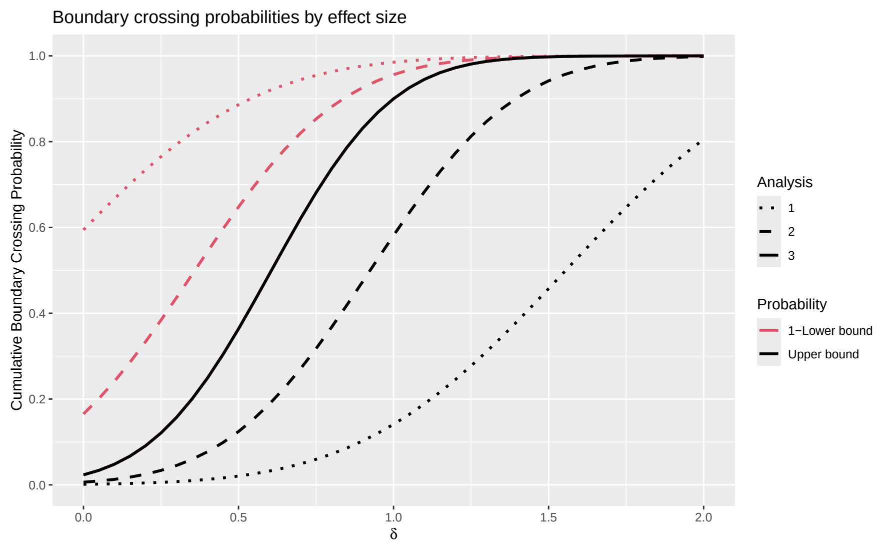
```
# Update time-to-event group sequential design
yu <- gsDesign(
  # Number of analyses
  k = 3,
  # Type I error
  alpha = y$alpha,
  # Type II error (1 - power)
  beta = y$beta,
  # Final planned event count
  maxn.IPlan = max(y$n.I),
  # Actual event counts at analyses
  n.I = c(125, 250, 364),
  # Bound type (in this case, non-binding futility)
  test.type = y$test.type,
  # Efficacy spending function
  sfu = y$upper$sf,
  # Parameter(s) for beta-spending (futility)
  sfupar = y$upper$param,
  # Futility spending function
  sfl = y$lower$sf,
  # Futility spending function parameter(s)
  sflpar = y$lower$param,
  # Standardized effect size
  delta = y$delta,
  # Natural parameter under H1 (ln(HR))
  delta1 = y$delta1,
  # Natural parameter under H0 (ln(1) = 0 for superiority)
  delta0 = y$delta0
)
gsBoundSummary(
  yu,
  deltaname = "HR",
  logdelta = TRUE,
  Nname = "Events",
  digits = 4,
  exclude = c(
    "Z", "B-value", "CP", "CP H1", "PP", "~HR at bound",
    # Following line deletes probability of boundary crossing
    # under null and alternate hypothesis; this must be customized,
    # if needed
    paste0("P(Cross) if HR=", c("0.7", "1"))
  )
)
#>     Analysis               Value Efficacy Futility
#>    IA 1: 36%         p (1-sided)   0.0015   0.5565
#>  Events: 125            Spending   0.0015   0.0162
#>                P(Cross) if HR=2.72   0.1641   0.0162
#>    IA 2: 71%         p (1-sided)   0.0066   0.1381
#>  Events: 250            Spending   0.0061   0.0329
#>                P(Cross) if HR=2.72   0.6414   0.0491
#>        Final         p (1-sided)   0.0223   0.0223
#>  Events: 364            Spending   0.0175   0.0509
#>                P(Cross) if HR=2.72   0.9046   0.0954
```

To avoid confusion we have limited the number of parameters displayed as only the 1-sided nominal *p*-value (`p (1-sided)`) should be needed to evaluate boundary crossing. Different parameters can be excluded from the table by changing the list passed to the `exclude` argument.

4.6 Further properties of designs

We consider further the design for a time-to-event variable assuming proportional hazards. To understand properties of the design better, we consider selected summary plots and we also demonstrate the `gsProbability()` function. First, we plot the operating characteristics for a larger set of hazard ratios than in the standard printout above.

```
plot(y, plottype = "Power")
```



The solid black line is of most interest as it shows the power of the design for different underlying hazard ratios. The dotted black line shows the probability of crossing the efficacy bound at the first interim, while the dashed black line shows the cumulative probability of crossing at the first or second interim. The red dotted line shows 1 minus the probability of crossing the futility bound at the first interim analysis for different hazard ratios. The red dashed line show 1 minus the cumulative probability of crossing the futility bound at the first or second interim analysis. For each hazard ratio, the lines divide the outcomes vertically into mutually exclusive possibilities for the trial; in this case (from bottom to top) 1) probability of first cross being efficacy at interim 1 (below dotted black line), 2) first cross being efficacy at interim 2 (between dotted and dashed black lines), 3) first crossing being efficacy at final analysis (between dashed and solid black lines), 4) first crossing being futility at final analysis (between solid black and dashed red lines), 5) first crossing being futility at analysis 2 (between dashed and dotted red lines), and 6) first crossing being futility at analysis 1 (above red dotted red line).

Note that the `plot()` function for the `gsDesign` class (used here) is an extension of the standard `plot()` function, and thus allows use of many of its parameters, such as line width (`lwd`), line type (`lty`), plot titles and axis labels.

The first two lines of code below demonstrate that a group sequential design generated by `gsDesign()` or `gsSurv()` can be input to `gsProbability()` to obtain boundary crossing probabilities for an extended set of parameter values. The `theta` values in the output make more sense in this case when they are computed relative to the effect size (log hazard ratio) `y$delta1=` 1 or hazard ratio `exp(y$delta1)=` 2.72for which the trial is powered; this is more easily seen in the power plot above than in the printout below.

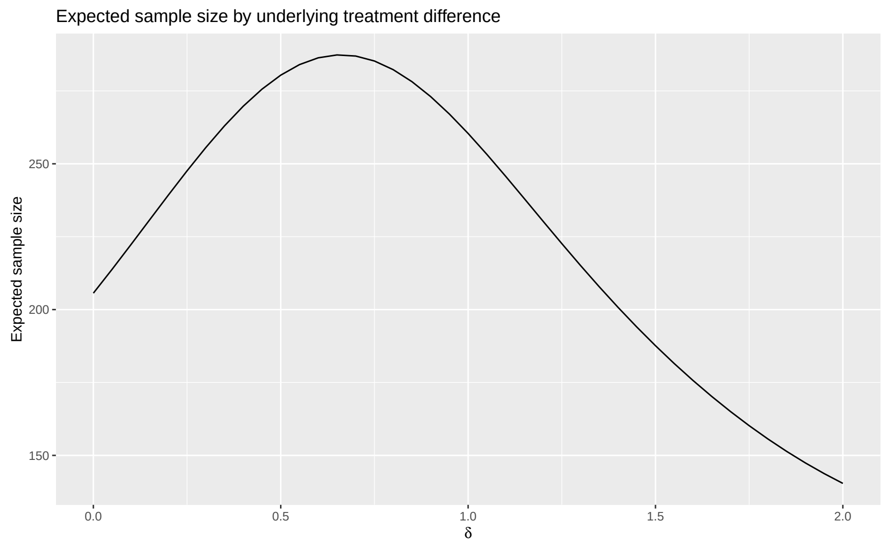
```
hr <- seq(0.4, 1.1, 0.1)
yp <- gsProbability(theta = log(hr) * y$delta / y$delta1, d = y)
yp
#> Asymmetric two-sided group sequential design with
#> 90 % power and 2.5 % Type I Error.
#> Upper bound spending computations assume
#> trial continues if lower bound is crossed.
#>
#>                  ----Lower bounds----  ----Upper bounds-----
#>   Analysis   N     Z   Nominal p Spend+  Z   Nominal p Spend++
#>          1 118 -0.24    0.4057 0.0148 3.01    0.0013  0.0013
#>          2 235  0.94    0.8267 0.0289 2.55    0.0054  0.0049
#>          3 353  2.00    0.9772 0.0563 2.00    0.0228  0.0188
#>      Total                     0.1000                 0.0250
#> + lower bound beta spending (under H1):
#>  Hwang-Shih-DeCani spending function with gamma = -2.
#> ++ alpha spending:
#>  Hwang-Shih-DeCani spending function with gamma = -4.
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>     Theta      1      2      3  Total  E{N}
#>   -0.1637 0.0000 0.0000 0.0000 0.0000 124.7
#>   -0.1239 0.0000 0.0000 0.0000 0.0000 133.5
#>   -0.0913 0.0000 0.0000 0.0001 0.0001 145.0
#>   -0.0637 0.0001 0.0002 0.0005 0.0008 158.6
#>   -0.0399 0.0003 0.0007 0.0023 0.0033 173.6
#>   -0.0188 0.0007 0.0021 0.0071 0.0098 189.5
#>    0.0000 0.0013 0.0049 0.0171 0.0233 205.6
#>    0.0170 0.0024 0.0101 0.0343 0.0467 221.4
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>     Theta      1      2      3  Total
#>   -0.1637 0.9376 0.0621 0.0003 1.0000
#>   -0.1239 0.8650 0.1329 0.0021 1.0000
#>   -0.0913 0.7734 0.2175 0.0089 0.9999
#>   -0.0637 0.6743 0.2997 0.0252 0.9992
#>   -0.0399 0.5766 0.3662 0.0538 0.9967
#>   -0.0188 0.4861 0.4098 0.0942 0.9902
#>    0.0000 0.4057 0.4290 0.1420 0.9767
#>    0.0170 0.3361 0.4264 0.1909 0.9533
```

Say we wish to compare values for `y$theta = 0.2559` to the plot above; we can translate to the HR scale with

```
exp(yp$theta[3] * y$delta1 / y$delta)
#> [1] 0.6
```

The we see the incremental probabilities described for the plot when HR = 0.6 are (from the table above): 0.4029, 0.5141, 0.0787, 0.0018, 0.0012, and 0.0013, respectively. Thus, the most likely outcomes are positive efficacy findings at interim 1 or 2 when the true underlying HR=0.6.

```
p <- plot(y, plottype = 6)
p
```



Following is a plot with the approximate hazard ratio required to cross each bound. The y-axis shows the expected number of events in the analysis where a bound is first crossed or, if no bound is crossed, the final analysis. relative to a fixed design trial when the sample size ratio is computed; if we had input a fixed design sample size, the y-axis would show the actual expected sample size. This plot demonstrates the ability of a group sequential design to appropriate adapt sample size to come to an appropriate conclusion depending on the true treatment effect.

```r
p <- plot(y, plottype = "HR")
p
```

Chapter 5

Time-to-event sample size derivation

We extend the Lachin and Foulkes [Lachin and Foulkes, 1986] method to cases where the null hypothesis does not reflect equality. This includes non-inferiority scenarios. For vaccines or other prevention studies this also includes super-superiority. Denote the null hypothesis failure rates for control and experimental treatment groups as λ_{00} and λ_{01}, respectively. Denote the alternate hypothesis rates as λ_{10} and λ_{11}. Further, denote the alternate hypothesis hazard ratio $h_1 = \lambda_{11}/\lambda_{10}$, and the null hypothesis hazard ratio $h_0 = \lambda_{01}/\lambda_{00}$. We let censoring rates be specific to the control (η_0) and experimental (η_1) groups; these values are only implicit in the equations below. Further, we let ξ denote the proportion of subjects randomized to the experimental treatment group. Finally, we let η represent an exponential dropout rate independent and the time to dropout is independent of the time until an event. Lachin and Foulkes assumed a null hypothesis with no difference between failure rates in the control and experimental rates and test for superiority. That is, $\lambda_{00} = \lambda_{01}$ ($h_0 = 1$) and $\lambda_{10} < \lambda_{01}$ ($h_1 < 1$). They set event rates under the null hypothesis so that the the weighted average event rate is the same under the null and alternate hypotheses:

$$\lambda_{00} = \lambda_{01} = \bar{\lambda} = (1 - \xi)\lambda_{10} + \xi\lambda_{11}. \tag{5.1}$$

The apparent intent of this is to equalize the variance for the log hazard ratio under null and alternative hypotheses; this will not be exactly the case. We let δ represent an indicator that an uncensored event is observed for a patient in a specified treatment group given enrollment, event rate, and dropout rate assumptions. The Lachin and Foulkes power equation for proportional hazards translates in our notation to:

$$\begin{aligned}\sqrt{N}\ln(h_1) &= Z_\alpha\sqrt{E\left\{\delta|\bar{\lambda},\eta\right\}^{-1}(\xi^{-1}+(1-\xi)^{-1})} \\ &+ Z_\beta\sqrt{E\left\{\delta|\lambda_1,\eta\right\}^{-1}\xi^{-1}+E\left\{\delta|\lambda_0,\eta\right\}^{-1}(1-\xi)^{-1}}\end{aligned} \tag{5.2}$$

Lachin and Foulkes did not cover any cases other than equality under the null hypothesis; i.e., the assumed $h_0 \neq 1$ (i.e., $\lambda_{00} \neq \lambda_{01}$). Equation 5.2 generalizes in this case to

$$\begin{aligned}\sqrt{N}\ln\left(\frac{h_1}{h_0}\right) &= Z_\alpha\sqrt{E\left\{\delta|\lambda_{01},\eta_1\right\}^{-1}\xi^{-1}+E\left\{\delta|\lambda_{00},\eta_0\right\}^{-1}(1-\xi)^{-1}} \\ &+ Z_\beta\sqrt{E\left\{\delta|\lambda_{11},\eta_1\right\}^{-1}\xi^{-1}+E\left\{\delta|\lambda_{10},\eta_0\right\}^{-1}(1-\xi)^{-1}}\end{aligned} \tag{5.3}$$

While we have defined null hypothesis assumptions λ_{00} and λ_{01} for an exponential distribution, the gsDesign functions `nSurv()` and `gsSurv()` extend the approach above in an analogous fashion to piecewise exponential failure and dropout rates with a common proportional hazard ratio across piecewise intervals.

For a fixed sample size with default arguments, we have:

```r
library(gsDesign)

fixed_design <- nSurv()
fixed_design
#> Fixed design, two-arm trial with time-to-event
#> outcome (Lachin and Foulkes, 1986).
#> Solving for:  Accrual rate
#> Hazard ratio                  H1/H0=0.6/1
#> Study duration:                   T=18
#> Accrual duration:                   12
#> Min. end-of-study follow-up: minfup=6
#> Expected events (total, H1):        160.4832
#> Expected sample size (total):       250.4492
#> Accrual rates:
#>      Stratum 1
#> 0-12   20.8708
#> Control event rates (H1):
#>       Stratum 1
#> 0-Inf    0.1155
#> Censoring rates:
#>       Stratum 1
#> 0-Inf         0
#> Power:                 100*(1-beta)=90%
#> Type I error (1-sided):   100*alpha=2.5%
#> Equal randomization:          ratio=1
```

This intentionally does not round up, so the user needs to round the number of events and sample size up. For `gsSurv()`, this rounding can be done automatically:

```
gs_design <- gsSurv() |> toInteger()
gs_design |> gsBoundSummary()
#>     Analysis              Value Efficacy Futility
#>    IA 1: 33%                  Z   3.0139  -0.2458
#>       N: 190        p (1-sided)   0.0013   0.5971
#>   Events: 57        ~HR at bound   0.4500   1.0673
#>     Month: 8   P(Cross) if HR=1   0.0013   0.4029
#>              P(Cross) if HR=0.6   0.1396   0.0147
#>    IA 2: 66%                  Z   2.5528   0.9301
#>       N: 268        p (1-sided)   0.0053   0.1762
#>  Events: 114        ~HR at bound   0.6199   0.8401
#>    Month: 13   P(Cross) if HR=1   0.0061   0.8320
#>              P(Cross) if HR=0.6   0.5769   0.0433
#>        Final                  Z   1.9988   1.9988
#>       N: 268        p (1-sided)   0.0228   0.0228
#>  Events: 172        ~HR at bound   0.7373   0.7373
#>    Month: 18   P(Cross) if HR=1   0.0233   0.9767
#>              P(Cross) if HR=0.6   0.9005   0.0995
```

A textual summary is also available:

```
summary(gs_design)
```

```
#> Asymmetric two-sided group sequential design with
#> non-binding futility bound, 3 analyses, time-to-event
#> outcome with sample size 268 and 172 events required, 90
#> percent power, 2.5 percent (1-sided) Type I error to detect
#> a hazard ratio of 0.6. Enrollment and total study durations
#> are assumed to be 12 and 18 months, respectively. Efficacy
#> bounds derived using a Hwang-Shih-DeCani spending function
#> with gamma = -4. Futility bounds derived using a
#> Hwang-Shih-DeCani spending function with gamma = -2.
```

All the assumptions laid out in this text can be changed as documented in the help file.

Chapter 6

Deriving group sequential designs

There are many ways to specify a group sequential design to obtain a desired power and Type I error. For planning purposes, the number, k and relative timing $0 < t_1 < \cdots < t_k = 1$ of interim analyses are fixed. Given these values, there are two general approaches to deriving boundaries for a group sequential trial:

- **The error spending approach**. Specify boundary crossing probabilities at each analysis and derive a sample size and boundary values based on these values. This is most commonly done with the error spending function approach proposed by Lan and DeMets [Lan and DeMets, 1983], which is discussed at some length in Chapter 8. We present this method in brief in Section 6.1 and follow this with simple examples.
- **The boundary family approach.** Specify how big boundary values should be relative to each other and adjust these relative values by a constant multiple to control overall error rates. Sample size adjustment is also part of this derivation. The commonly applied boundary family approach uses the Wang-Tsiatis [Wang and Tsiatis, 1987] family which includes bounds by Pocock [Pocock, 1977] and O'Brien and Fleming [O'Brien and Fleming, 1979]. This will be discussed in Section 6.2.

6.1 Boundary derivation using boundary crossing probabilities

6.1.1 Types of error probabilities used: `test.type`

Before starting a discussion of spending functions, different methods of computing Type I error are discussed. Boundary crossing probabilities for upper bounds may be specified to `gsDesign()` using either $\alpha_i(0)$ or $\alpha_i^+(0)$,

$i = 1, 2, \ldots, k$. In the first case, it is assumed that a trial stops when either a lower or upper bound is crossed and the only Type I error occurs when the first time a boundary is crossed it is an upper bound. In the second case, it is assumed that a lower bound may be ignored if crossed and the Type I error is the probability of ever crossing an upper bound if the trial is never stopped for crossing a lower bound. As we have seen, the difference between these boundary crossing probabilities may be small. The differences can be meaningful, however, when aggressive futility bounds are employed to require, say, an early positive treatment effect trend.

For lower bounds, either $\beta_i(\delta)$ or $\beta_i(0)$, $i = 1, 2, \ldots, k$, may be specified.

Sample size and boundaries that have appropriate boundary crossing probabilities and power are derived numerically using computational methods given in detail in Chapter 19 of Jennison and Turnbull [Jennison and Turnbull, 2000]. The `gsDesign()` parameter `test.type` specifies which boundary crossing probabilities are used as outlined in Table 6.1.

Table 6.1: Boundary crossing probabilities used to set boundaries in `gsDesign()` by `test.type`.

test.type	Upper bound	Lower bound
1	$\alpha_i^+(0)$	None
2	$\alpha(0)$	$\beta_i(0)$
3	$\alpha_i(0)$	$\beta_i(\delta)$
4	$\alpha_i^+(0)$	$\beta_i(\delta)$
5	$\alpha(0)$	$\beta_i(0)$
6	$\alpha^+(0)$	$\beta_i(0)$

For `test.type` $=$ 1, 2 and 5, boundaries can be computed in a single step just by knowing the cumulative proportion of the final planned statistical information (sample size/number of events) at each analysis that is specified using the `timing` input variable. For `test.type` $=$ 6, the upper and lower boundaries are computed separately and independently using these same methods. For `test.type` $=$ 1, 2, 5 or 6, the total sample size is then set to obtain the desired power under the alternative hypothesis by using a root finding algorithm.

For `test.type` $=$ 3 and 4, sample size and bounds are set simultaneously using an iterative algorithm. This computation is slightly more complex than the above. This does not make any noticeable difference in normal use of the `gsDesign()`. However, for user-developed routines that require repeated calls to `gsDesign()` (e.g., finding an optimal design), there may be noticeably slower performance when `test.type` $=$ 3 or 4 is used.

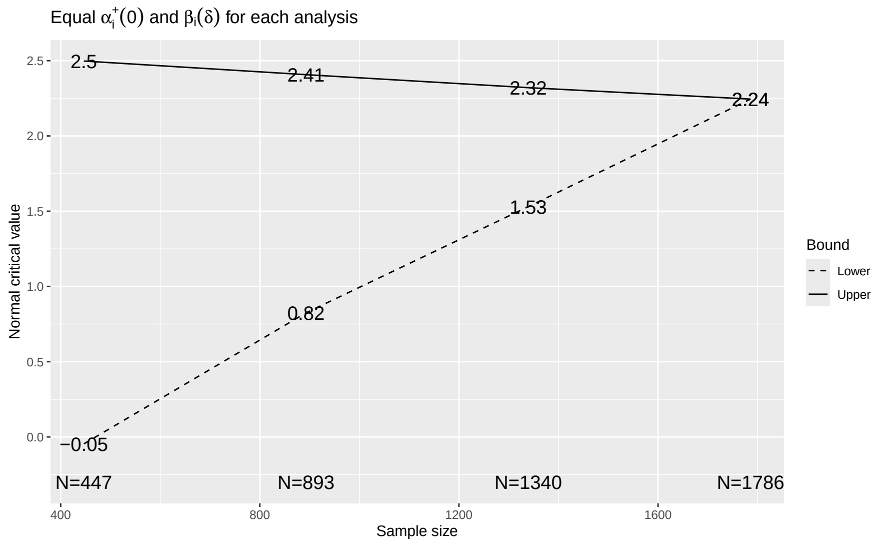
6.1.2 Specifying boundary crossing probabilities in `gsDesign()`

We use the CAPTURE example, working with the desired 80% power ($\beta = 0.2$) to demonstrate deriving bounds with specified boundary crossing probabilities. For simplicity, we will let $\alpha_i^+(0) = 0.025/4$ and $\beta_i(\delta) = 0.2/4$, $i = 1, 2, 3, 4$. Setting the `gsDesign()` parameters `sfu` and `sfl` to `sfLinear`, the vector `p` below is used to equally allocate the boundary crossing probabilities for each analysis. Note that `sfLinear()` requires an even number of elements in `param`. The first half specify the timepoints using an increasing set of values strictly between 0 and 1 to indicate the proportion of information at which the spending function is specified. The second half specify the proportion of total error spending at each of these points. (Aside: those interested in plotting with special characters note the special handling of the character + in the argument `main` to `plot()`.)

```r
library(gsDesign)
```

```r
# Cumulative proportion of spending planned at each analysis.
# In this case, this is also proportion of final observations
# at each interim.
p <- c(0.25, 0.5, 0.75)
t <- c(0.25, 0.5, 0.75)
# Cumulative spending intended at each analysis
# (for illustration)
p * 0.025
#> [1] 0.00625 0.01250 0.01875
```

```r
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
x <- gsDesign(
  k = 4,
  n.fix = n.fix,
  beta = 0.2,
  sfu = sfLinear,
  sfupar = c(t, p),
  sfl = sfLinear,
  sflpar = c(t, p)
)
plot(
  x,
  main = expression(
    paste(
      "Equal ",
      alpha[i]^{
        "+"
      }, (0),
      " and ",
      beta[i](delta),
      " for each analysis"
    )
  )
)
```

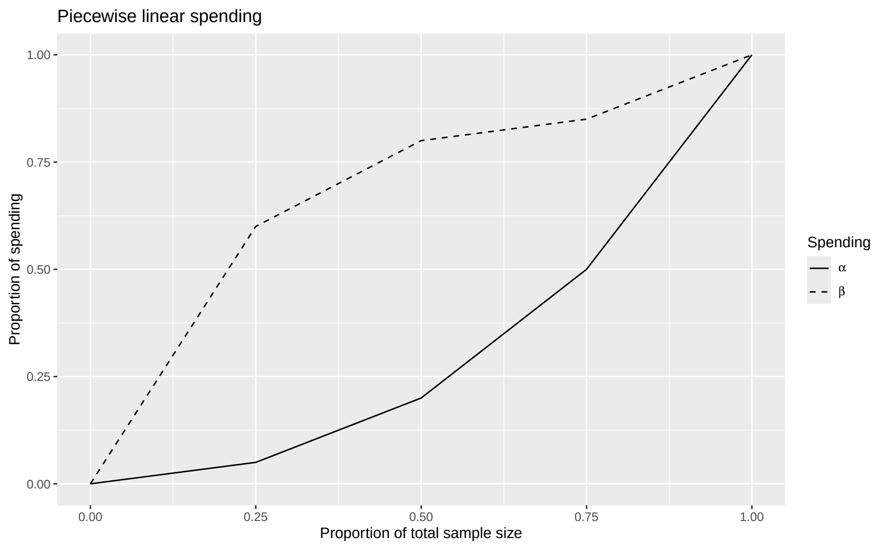


```
x
#> Asymmetric two-sided group sequential design with
#> 80 % power and 2.5 % Type I Error.
#> Upper bound spending computations assume
#> trial continues if lower bound is crossed.
#>
#>                   ----Lower bounds----  ----Upper bounds-----
#>   Analysis   N    Z   Nominal p Spend+  Z   Nominal p Spend++
#>          1  447 -0.05    0.4816   0.05 2.50    0.0063  0.0063
#>          2  893  0.82    0.7953   0.05 2.41    0.0080  0.0063
#>          3 1340  1.53    0.9370   0.05 2.32    0.0101  0.0063
#>          4 1786  2.24    0.9876   0.05 2.24    0.0124  0.0062
#>      Total                      0.2000                 0.0250
#> + lower bound beta spending (under H1):
#>  Piecewise linear spending function with line points = 0.25, line points = 0.5, line poi
#> ++ alpha spending:
#>  Piecewise linear spending function with line points = 0.25, line points = 0.5, line poi
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>    Theta      1      2      3      4  Total   E{N}
#>   0.0000 0.0063 0.0062 0.0059 0.0042 0.0225  769.7
#>   0.0757 0.1843 0.2805 0.2253 0.1100 0.8000 1054.1
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>    Theta      1      2      3      4  Total
#>   0.0000 0.4816 0.3321 0.1299 0.0339 0.9775
#>   0.0757 0.0500 0.0500 0.0500 0.0500 0.2000
```

The printed output from the above is shown below and a plot of the derived boundaries is in Figure below. The columns labeled `Spend+` and `Spend++` show the values $\beta_i(\delta)$ and $\alpha_i(0)$, respectively, are equal for each analysis, $i = 1, 2, 3, 4$. The nominal p-values for the upper bound increase and thus the bounds themselves decrease for each analysis. That equal error probabilities results in unequal bounds is because of the correlation between the test statistics used for analysis that was indicated in Section 2.1. Note that the requirement of 1372 patients for the fixed design has now increased to a maximum sample size of 1786 which is an inflation of 30%. On the other hand, the expected number of patients when a boundary is crossed is 770 under the assumption of no treatment difference and 1054 under the alternative hypothesis of a 15% event rate in the control group and 10% in the experimental group. Thus, this redesign seems reasonably effective at controlling the sample size when the experimental regimen has no underlying benefit. The nominal α—level of 0.0124 required for a positive result at the end of the study is almost exactly half that of the overall 0.025 for the study. We will propose other designs that will not require such a small final nominal α by setting higher early efficacy bounds.

Now we display piecewise linear spending functions. The plot resulting from the code below is displayed below.

```r
# Cumulative proportion of spending planned at each analysis
# Now use a piecewise linear spending
p <- c(0.05, 0.2, 0.5)
p2 <- c(0.6, 0.8, 0.85)
x <- gsDesign(
  k = 4,
  n.fix = n.fix,
  beta = 0.2,
  sfu = sfLinear,
  sfupar = c(t, p),
  sfl = sfLinear,
  sflpar = c(t, p2)
)
plot(x, plottype = "sf", main = "Piecewise linear spending")
```



6.2 Deriving group sequential designs using boundary families

The second method of setting boundaries uses the relative z-value for a design cutoff at each interim analysis, $c_i > 0$, $i = 1, 2, \ldots, k$. We define vectors $\mathbf{t} \equiv (t_1, t_2, \ldots, t_k)$ and $\mathbf{c} \equiv (c_1, c_2, \ldots, c_k)$. For 2-sided testing, Wang and Tsiatis [Wang and Tsiatis, 1987] defined the boundary function

$$-a_i = b_i = C(\mathbf{t}, \mathbf{c})c_i$$

where the constant $C(\mathbf{t}, \mathbf{c}) > 0$ is chosen to appropriately control Type I error.

Wang and Tsiatis [Wang and Tsiatis, 1987] specifically defined the boundary function family

$$g(t; \Delta) = C(\mathbf{t}; \Delta)t^{\Delta - 0.5}.$$

For $i = 1, 2, \ldots, k$, the boundary at analysis i are given by

$$-a_i = b_i = C(\mathbf{t}; \Delta) t_i^{\Delta - 0.5}.$$

For 2-sided testing, note that for $\Delta = 0.5$ the boundary values are all equal. Thus, this is equivalent to a Pocock [Pocock, 1977] boundary when analyses are equally spaced. The value $\Delta = 0$ generates O'Brien-Fleming bounds [O'Brien and Fleming, 1979].

Pampallona and Tsiatis [Pampallona and Tsiatis, 1994] derived a related method of using boundary families to set asymmetric bounds; this is not currently implemented in `gsDesign()`. Using constants $c_i' > 0$, $i = 1, 2, \ldots, k$ and a constant $C'(\mathbf{t}; \mathbf{I_k})$ that along with I_k is used to appropriately control Type II error, they set

$$a_i = \delta\sqrt{t_i} - C'(\mathbf{t}) c_i'.$$

O'Brien-Fleming, Pocock, or Wang-Tsiatis are normally used with equally-spaced analyses. They are used only with one-sided (`test.type=1`) and symmetric two-sided (`test.type=2`) designs. We will use the CAPTURE example, again with 80% power rather than the default of 90%. Notice that this requires specifying `beta = 0.2` in both `nBinomial()` and `gsDesign()`. O'Brien-Fleming, Pocock, or Wang-Tsiatis (parameter of 0.15) bounds for equally space analyses are generated using the parameters `sfu` and `sfupar` below. If you print the Pocock design (`xPocock`), you will see that the upper bounds are all equal and that the upper boundary crossing values $\alpha_i(0)$ printed in the `Spend` column decrease from 0.0091 for the first analysis to 0.0041 for the final analysis.

```r
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
xOF <- gsDesign(
  k = 4,
  test.type = 2,
  n.fix = n.fix,
  sfu = "OF",
  beta = 0.2
)
xPocock <- gsDesign(
  k = 4,
  test.type = 2,
  n.fix = n.fix,
  sfu = "Pocock",
  beta = 0.2
)
xWT <- gsDesign(
  k = 4,
  test.type = 2,
  n.fix = n.fix,
  sfu = "WT",
  sfupar = 0.15,
  beta = 0.2
)
```

The resulting sample sizes for these designs can be computed using

```
nOF <- 2 * ceiling(xOF$n.I[4] / 2)
nPocock <- 2 * ceiling(xPocock$n.I[4] / 2)
nWT <- 2 * ceiling(xWT$n.I[4] / 2)
```

We now present an example of how is fairly simple to produce custom plots using `gsDesign()` output and standard R plotting functions. The resulting output is in Figure below. If you are not familiar with R plotting, executing the following statements one at a time may be instructive. The call `help(plot)` and its ”See also” links (especially `par`) can be used to find explanations of parameters below. The `legend` call below particularly demonstrates a nice strength of R for printing Greek characters and subscripts in plots.

```
plot(
  xOF$n.I,
  xOF$upper$bound,
  xlim = c(300, 1800),
  ylim = c(1.5, 4.5),
  pch = "o",
  cex = 1.5,
  lwd = 2,
  type = "b",
  xlab = "N",
  ylab = "Normal critical value (upper bounds)",
  main = "N and upper bounds with Wang-Tsiatis designs"
)
lines(xPocock$n.I, xPocock$upper$bound, lty = 3, lwd = 2)
points(xPocock$n.I, xPocock$upper$bound, pch = "p", cex = 1.5)
lines(xWT$n.I, xWT$upper$bound, lty = 2, lwd = 2)
points(xWT$n.I, xWT$upper$bound, pch = "x", cex = 1.5)
legend(
  x = c(600, 1825),
  y = c(3.4, 4.5),
  lty = c(1, 2, 3),
  lwd = 2,
  pch = c("o", "x", "p"),
  cex = 1.5,
  legend = c(
    expression(paste(
      Delta, "=0.0, ", N[4],
      "=1404, (O'Brien-Fleming)"
    )),
    c(
      expression(paste(
        Delta, "=0.15, ", N[4], "=1430"
      )),
      c(expression(paste(
        Delta, "=0.50, ", N[4], "=1650, (Pocock)"
      )))
    )
  )
)
```

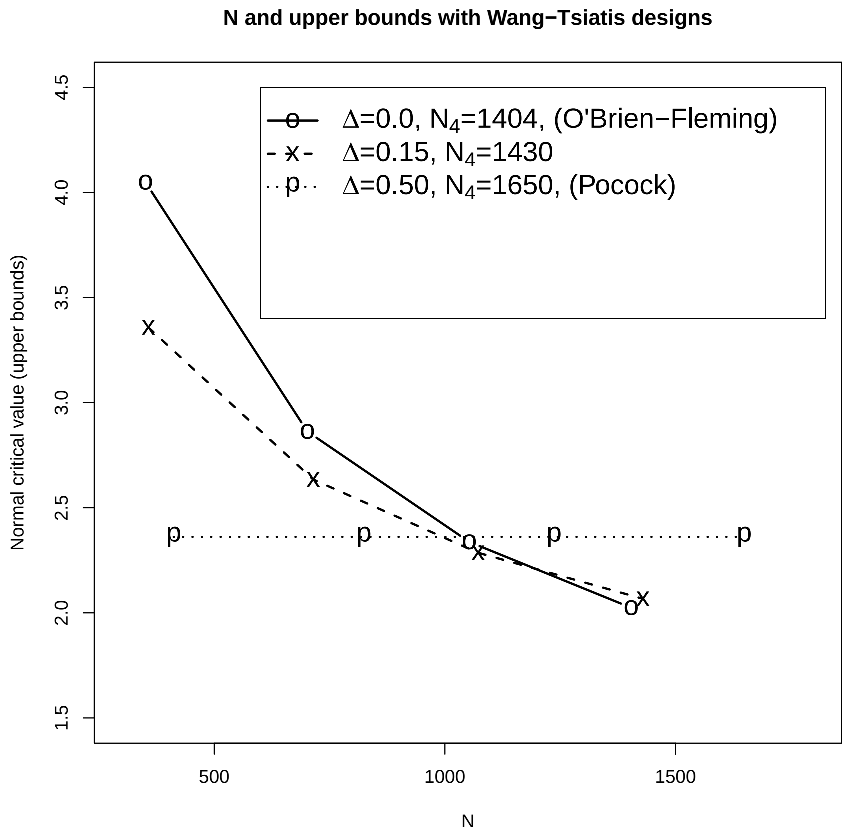


Figure 6.1

Figure 6.1 shows how the upper bounds and sample size change as Δ changes for Wang-Tsiatis bounds. For the O'Brien-Fleming design, the final sample size is only inflated to 1402 from the 1372 required for a fixed design. The relatively aggressive early bounds for the the Pocock design result in sample size inflation to 1650. This design is not frequently used because of the relatively low bounds at early analyses and the substantial sample size inflation required to maintain the desired power. Since the nominal p-value required for stopping at the initial analysis for the O'Brien-Fleming design is 0.00005 (2-sided), an intermediate design with $\Delta = 0.15$ might be of some interest. This has a relatively small sample size inflation to 1430 in order to maintain power and the nominal p-value required to stop the trial at the first interim analysis is 0.0008 (2-sided). Examine the boundary crossing probabilities by reviewing, for example, `xOF$upper$spend`. Also consider reviewing `plot(xWT, plottype = 3)` to see the observed treatment effect required at each analysis to cross a boundary.

Chapter 7

Other gsDesign() parameters

7.1 Setting Type I error and power

Type I error as input to `gsDesign()` is always one-sided and is set through the parameter `alpha`. Type II error (1-power) is set in the parameter `beta`. A standard design modified to have Type I error of 0.05 and Type II error of 0.2 (80% power) rather than the default of 0.025 Type I and 0.1 Type II error is produced with the command

```r
library(gsDesign)

x <- gsDesign(alpha = 0.05, beta = 0.2)
```

7.2 Number and timing of analyses

The number of analyses is set in `gsDesign()` through the parameter `k>1`, which has a default of 3. The default for timing of analyses is to have them equally-spaced, which is indicated by the default value of `timing=1`. This will often not be feasible or desired due to logistical or other reasons. The parameter `timing` can be input as a vector of length `k` or `k-1` where $0 <$ `timing[1]` $<$ `timing[2]` $< ... <$ `timing[k]` $= 1$. It is optional to specify `timing[k]` since it is always 1. The values in `timing` set the proportion of statistical information available for the data analyzed at each interim analysis. The statistical information is generally proportional to the number of observations analyzed or, for survival analysis, the number of time-to-event endpoints that have been observed. The following compares upper bounds, number of observations at each analysis, and average number of observations at the analysis where a boundary is crossed for the default design (stored in `x`) versus an alternative analyzing after 25%and 50% of observations (stored in `xt`) for the CAPTURE example. You can see that the upper bounds are more stringent when analyses are done earlier in the trial.

```
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1)
x <- gsDesign(n.fix = n.fix)
2 * ceiling(x$n.I / 2)
#> [1]  656 1310 1964
```

```
x$upper$bound
#> [1] 3.010739 2.546531 1.999226
```

```
x$en
#> [1] 1146.391 1451.709
```

```
xt <- gsDesign(n.fix = n.fix, timing = c(0.25, 0.5))
2 * ceiling(xt$n.I / 2)
#> [1]  482  964 1926
```

```
xt$upper$bound
#> [1] 3.155373 2.818347 1.983563
```

```
xt$en
#> [1] 1185.173 1547.649
```

Comparing the designs, we see that the average sample number is lower for the default design with evenly spaced analyses compared to the design analyzing after 25% and 50% of observations. This is true both under the null hypothesis (1146 versus 1185) and the alternate hypothesis (1452 versus 1548) in spite of a lower maximum sample size (1926 versus 1964) for the latter design. To understand this further we look first at the probability of crossing the lower bound at each analysis for each design below. The columns of the matrices printed correspond to the `theta` values under the null and alternate hypotheses, respectively, while rows correspond to the analyses. Thus, the default design has probability of 41% of crossing the lower bound at the first interim analysis compared to 25% for the design with first analysis at 25% of observations. By examining these probabilities as well as corresponding upper boundary crossing probabilities (e.g., `x$upper$prob`) we see that by moving analyses earlier without changing spending functions we have decreased the probability of crossing an interim boundary, which explains the smaller expected sample size for the default design which uses later interim analyses.

```
x$lower$prob
#>           [,1]       [,2]
#> [1,] 0.4056598 0.01483371
#> [2,] 0.4290045 0.02889212
#> [3,] 0.1420312 0.05627417
```

```
xt$lower$prob
#>           [,1]       [,2]
#> [1,] 0.2546094 0.01015363
#> [2,] 0.3839157 0.01674051
#> [3,] 0.3375615 0.07310586
```

7.3 Standardized treatment effect: `delta`

7.3.1 Normally distributed data

The "usual" formula for sample size for a fixed design is

$$n = \left(\frac{Z_{1-\alpha} + Z_{1-\beta}}{\delta}\right)^2. \tag{7.1}$$

This formula originates from testing the mean of a sample of normal random variables with variance 1. The null hypothesis is that the true mean θ equals 0, while the alternate hypothesis is that $\theta = \delta$. The distribution of the mean of n observations $\bar{X}_n$ follows a normal distribution with mean δ and standard deviation $1/n$ (i.e., N(δ,$1/n$)). Assuming $\delta > 0$, the standard statistic for testing this is $Z_n = \sqrt{n}\bar{X}_n$ ~N($\sqrt{n}\delta, 1$) which rejects the hypothesis that the true mean is 0 if $Z_n > Z_{1-\alpha}$. The null hypothesis is rejected with probability α when the null hypothesis is true (Type I error), while the probability of rejecting under the alternate hypothesis (power or one minus Type II error) is

$$\Phi(Z_{1-\alpha} - \sqrt{n}\delta). \tag{7.2}$$

By fixing this probability as $1 - \beta$ and solving for n, Equation 7.1 is derived.

Assume a set of patients is evaluated at baseline for a measure of interest, then treated with a therapy and subsequently measured for a change from baseline. Assume the within subject variance for the change from baseline is 1. Suppose $\delta = 0.1$. The default group sequential design can be obtained for such a study using the call `gsDesign(delta = 0.1)`, yielding a planned maximum sample size of 1125.

7.3.2 Time to event data

Equation 7.1 and Equation 7.2 are used as approximations for many situations where test statistics are approximated well by the normal distribution as n gets large. A useful example of this approximation is comparing survival distributions for two groups under the assumption that the hazard rate ("instantaneous failure rate") for the control group ($\lambda_1(t)$) and experimental group ($\lambda_2(t)$) for any time $t > 0$ are proportional as expressed by

$$\lambda_2(t) = e^{-\gamma}\lambda_1(t).$$

We have used $-\gamma$ in the exponent so that a positive value of γ indicates lower risk in the experimental treatment group. The value $e^{-\gamma}$ will be referred to as the hazard ratio and γ as minus the log hazard ratio.

Note that when $\gamma = 0$ there is no difference in the hazard rates between treatment groups. A common test statistic for the null hypothesis that $\gamma = 0$ is the logrank test. We will denote this by $T(d)$ where d indicates the number of events observed in the combined treatment groups. A reasonably good approximation for its distribution is

$$T(d) \sim \mathrm{N}(\gamma \times V(d), V(d)).$$

For equally sized treatment groups, $V(d)$ is approximately $d/4$. Thus,

$$Z = T(d)2/\sqrt{d} \sim \mathrm{N}(\sqrt{d}\gamma/2, 1).$$

For the formulation from Section 2.1 we have $\theta = \gamma/2$. If $\gamma = \mu$ is the alternative hypothesis to the null hypothesis $\gamma = 0$, then we have $\delta = \mu/2$. In fact, Tsiatis [Tsiatis, 1982], Sellke and Siegmund [Sellke and Siegmund, 1983] and Slud and Wei [Slud and Wei, 1982] have all shown that group sequential theory may be applied to censored survival data using this distributional assumption; this is also discussed by Jennison and Turnbull [Jennison and Turnbull, 2000]. If we assume there are k analyses after $d_1 < d_2 < \ldots < d_k$ events and let $\mathcal{I}_i = d_i/4$, $i = 1, 2, \ldots, k$ then we may apply the canonical distribution assumptions from Equation 2.7 and Equation 2.8.

For the cancer trial example in Section 1.6, we assumed $e^{-\mu} = 0.7$ which yields $\delta = -\ln(0.7)/2 = 0.178$. Applying Equation 7.1 with $\alpha = 0.025$ and $\beta = 0.1$ and this value of δ, the number of events required is calculated as 331 compared to 330 calculated previously using the Lachin and Foulkes [Lachin and Foulkes, 1986] method. We may obtain the default group sequential design by specifying δ rather than the fixed design number of events as follows: `gsDesign(delta = -log(0.7) / 2)`.

We also apply this distribution theory to the non-inferiority trial for a new treatment for diabetes. We wish to rule out a hazard ratio of 1.3 for the experimental group compared to the control group under the assumption that the risk of cardiovascular events is equal in the two treatment groups. This implies that our null hypothesis is that $\gamma = \ln(1.3) = 0.262$ and the alternate hypothesis is that $\gamma = 0$. Letting $\theta = (\gamma - \ln(1.3))/2$ the null hypothesis is re-framed as $\theta = 0$ and the alternative as $\theta = \ln(1.3)/2$. The test statistic $Z = (T(d) - \ln(1.3) \times d/4) \times 2/\sqrt{d}$ is then approximately distributed $\mathrm{N}(\sqrt{d}\theta, 1)$. Substituting $\delta = \ln(1.3)/2$, $\alpha = 0.025$ and $\beta = 0.1$ in Equation 7.1 we come up with $d = 611$. This is within 1% of the 617 events suggested in Section 1.8.

Chapter 8

Spending functions

8.1 Spending function definitions

For any given $0 < \alpha < 1$, we define a non-decreasing function $f(t;\alpha)$ for $t \geq 0$ with $\alpha(0) = 0$ and for $t \geq 1$, $f(t;\alpha) = \alpha$. For $i = 1, 2, \ldots, K$, we define $t_i = I_i/I_K$ and then set $\alpha_i(0)$ through the equation

$$f(t_i;\alpha) = \sum_{j=1}^{i} \alpha_j(0).$$

We consider a spending function proposed by Lan and DeMets [Lan and DeMets, 1983] to approximate a Pocock bound.

$$f(t;\alpha) = \alpha \log(1 + (e-1)t)$$

This spending function is implemented in the function `sfLDPocock`. We again consider a 2-sided design with equally spaced analyses, $t_i = i/4$, $i = 1, 2, 3, 4$. The values for $\alpha_i(0)$ are obtained as follows:

```
library(gsDesign)

sfLDPocock(alpha = 0.025, t = 1:4 / 4)$spend
#> [1] 0.00893435 0.01550286 0.02069972 0.02500000
```

We will discuss the exact nature of this call to `sfLDPocock` in Section 8.5 below. We now derive a design for the CAPTURE study using this spending function

```
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
x <- gsDesign(
  k = 4,
```

```r
  test.type = 2,
  n.fix = n.fix,
  sfu = sfLDPocock,
  beta = 0.2
)
cumsum(x$upper$prob[, 1])
#> [1] 0.00893435 0.01550287 0.02069973 0.02500001
```

The boundary crossing probabilities under the assumption of no treatment difference are in `x$upper$prob[,1]` and there cumulative totals are produced by the above call to `cumsum()`. Note that these values match those produced by the call to `sfLDPocock` above. Next we compare the bounds produced by this design with the actual Pocock bounds to see they are nearly identical:

```r
xPocock <- gsDesign(
  k = 4,
  test.type = 2,
  n.fix = n.fix,
  sfu = "Pocock",
  beta = 0.2
)
x$upper$bound
#> [1] 2.368328 2.367524 2.358168 2.350030
```

```r
xPocock$upper$bound
#> [1] 2.361298 2.361298 2.361298 2.361298
```

The reader may wish to compare the O'Brien-Fleming design presented in Section 6.2 using the spending function `sfLDOF`, which implements a spending function proposed by Lan and DeMets [Lan and DeMets, 1983] to approximate this design:

$$\alpha_i(t) = 2\left(1 - \Phi\left(\frac{\Phi^{-1}(\alpha/2)}{\sqrt{t}}\right)\right)$$

You will see that this approximation is not as good as the Pocock bound approximation.

8.2 Spending function families

The function $f(t;\alpha)$ may be generalized to a family $f(t;\alpha,\gamma)$ of spending functions using one or more parameters. For instance, the default Hwang-Shih-DeCani spending function family is defined for $0 \leq t \leq 1$ and any real γ by

$$f(t;\alpha,\gamma) = \begin{array}{ll} \alpha\frac{1-\exp(-\gamma t)}{1-\exp(-\gamma)}, & \gamma \neq 0 \\ \alpha t, & \gamma = 0 \end{array}$$

The boundary crossing probabilities $\alpha_i^+(\theta)$ and $\beta_i(\theta)$ may be defined in a similar fashion, $i = 1, 2, \ldots, K$ with the same or different spending functions f where:

$$f(t_i;\alpha) = \sum_{j=1}^{i} \alpha_j^+(0) \tag{8.1}$$

$$f(t_i;\beta(\theta)) = \sum_{j=1}^{i} \beta_j(\theta) \tag{8.2}$$

The argument `test.type` in `gsDesign()` provides two options for how to use $f(t_i;\beta(\theta))$ to set lower bounds. For `test.type`=2, 5 and 6, lower boundary values are set under the null hypothesis by specifying $\beta(t;0)$, $0 \leq t$. For `test.type`=3 and 4, we compute lower boundary values under the alternative hypothesis by specifying $\beta(t;\delta)$, $0 \leq t$. $\beta(t;\delta)$ is referred to as the β-spending function and the value $\beta_i(\delta)$ is referred to as the amount of β (Type II error rate) spent at analysis i, $1 \leq i \leq K$.

Standard published spending functions commonly used for group sequential design are included as part of the gsDesign package. Several 'new' spending functions are included that are of potential interest. Users can also write their own spending functions to pass directly to `gsDesign()`. Available spending functions and the syntax for writing a new spending function are documented here. We begin here with simple examples of how to apply standard spending functions in calls to `gsDesign()`. This may be as far as many readers may want to read. However, for those interested in more esoteric spending functions, full documentation of the extensive spending function capabilities available is included. Examples for each type of spending function in the package are included in the online help documentation.

8.3 Spending function basics

The parameters `sfu` and `sfl` are used to set spending functions for the upper and lower bounds, respectively, each having a default value of `sfHSD`, the Hwang-Shih-DeCani spending function. The default parameter for the upper bound is $\gamma = -4$ to produce a conservative, O'Brien-Fleming-like bound. The default for the lower bound is $\gamma = -2$, a less conservative bound. This design was presented at some length in Section 4.1.

To change these to -3 (less conservative than an O'Brien-Fleming bound) and 1 (an aggressive Pocock-like bound), respectively, requires the parameter `sfupar` for the upper bound and `sflpar` for the lower bound: Next we consider some simple alternatives to the default spending function parameters. The Kim-DeMets function, `sfPower()`, with upper bound parameter $\rho = 3$ (a conservative, O'Brien-Fleming-like bound) and lower bound parameter $\rho = 0.75$ (an aggressive, Pocock-like bound) requires resetting the upper bound spending function `sfu` and the lower bound spending function `sfl`. In the first code line following, we replace lower and upper spending function parameters with 1 and -2, respectively; the default Hwang-Shih-DeCani spending function family is still used. In the second line, we specify a Kim-DeMets (power) spending function for both the lower bound (with the parameters `sfl=sfPower` and `sflpar=2`) and the upper bounds (with the parameters `sfu=sfPower` and `sfupar=3`). Then we compare bounds from the three designs. Bounds for the power spending function design are quite comparable to the default design. Generally, choosing between these two spending function families is somewhat arbitrary. The alternate Hwang-Shih-DeCani design uses more aggressive stopping boundaries. The last lines below show that sample size inflation from a fixed design is about 25% for the the design with more aggressive stopping boundaries compared to about 7% for each of the other designs.

```
x <- gsDesign()
xHSDalt <- gsDesign(
  sflpar = 1,
  sfupar = -2
)
xKD <- gsDesign(
  sfl = sfPower,
  sflpar = 2,
  sfu = sfPower,
  sfupar = 3
)
x$upper$bound
#> [1] 3.010739 2.546531 1.999226
```

```
xHSDalt$upper$bound
#> [1] 2.677524 2.385418 2.063740
```

```
xKD$upper$bound
#> [1] 3.113017 2.461933 2.008705
```

```
x$lower$bound
#> [1] -0.2387240  0.9410673  1.9992264
```

```
xHSDalt$lower$bound
#> [1] 0.3989131 1.3302942 2.0637399
```

```
xKD$lower$bound
#> [1] -0.3497490  0.9822542  2.0087052
```

```
x$n.I[3]
#> [1] 1.069883
```

```
xHSDalt$n.I[3]
#> [1] 1.254268
```

```
xKD$n.I[3]
#> [1] 1.071011
```

Following is example code to plot Hwang-Shih-DeCani spending functions for three values of the γ parameter. The first two γ values are the defaults for upper bound spending ($\gamma = -4$; a conservative bound somewhat similar to an O'Brien-Fleming bound) and lower bound spending ($\gamma = -2$; a less conservative bound). The third ($\gamma = 1$) is included as it approximates a Pocock stopping rule; see Hwang, Shih and DeCani [Hwang et al., 1990]. The Hwang-Shih-DeCani spending function class implemented in the function `sfHSD()` may be sufficient for designing many clinical trials without considering the other spending function forms available in this package. The three parameters in the calls to `sfHSD()` below are the total Type I error, values for which the spending function is evaluated (and later plotted), and the γ parameter for the Hwang-Shih-DeCani design. The code below yields the plot in Figure 8.1 (note the typesetting of Greek characters!).

```
plot(0:100 / 100, sfHSD(0.025, 0:100 / 100, -4)$spend,
  type = "l", lwd = 2,
  xlab = "Proportion of information",
  ylab = expression(paste("Cumulative \ ", alpha, "-spending")),
  main = "Hwang-Shih-DeCani Spending Functions"
)
lines(
  0:100 / 100,
  sfHSD(0.025, 0:100 / 100, -2)$spend,
  lty = 2,
  lwd = 2
)
lines(
  0:100 / 100,
  sfHSD(0.025, 0:100 / 100, 1)$spend,
  lty = 3,
  lwd = 2
)
```

```r
legend(
  x = c(0.0, 0.27), y = 0.025 * c(0.8, 1), lty = 1:3, lwd = 2,
  legend = c(
    expression(paste(gamma, " = -4")),
    expression(paste(gamma, " = -2")),
    expression(paste(gamma, " = 1"))
  )
)
```

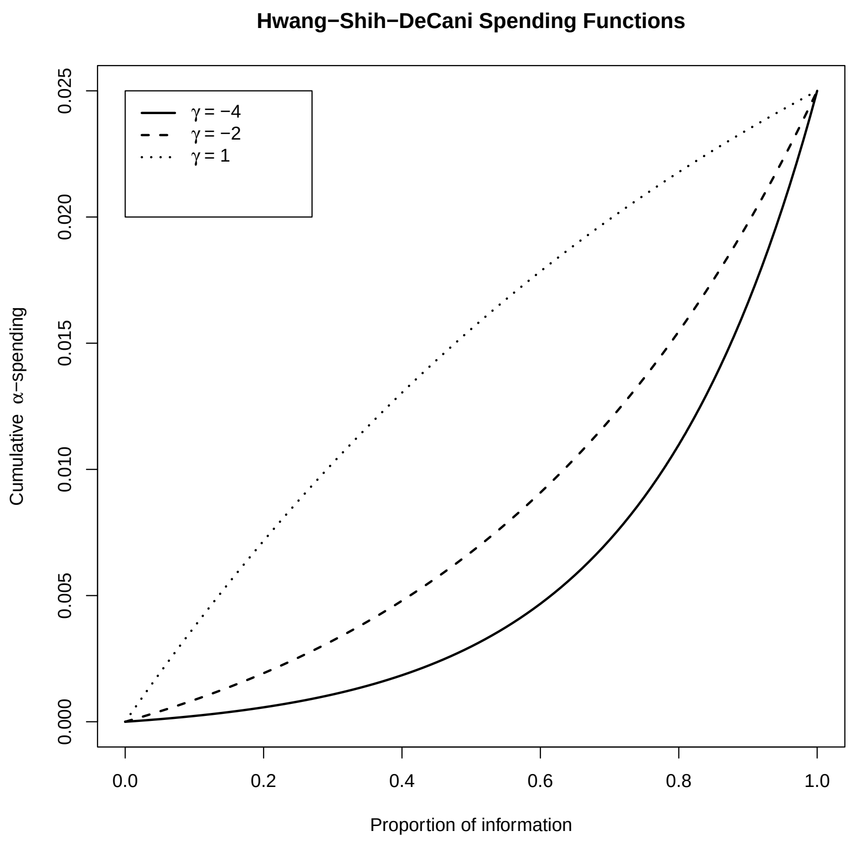


Figure 8.1

Similarly, Jennison and Turnbull [Jennison and Turnbull, 2000], suggest that the Kim-DeMets spending function is flexible enough to suit most purposes. To compare the Kim-DeMets family with the Hwang-Shih-DeCani family just demonstrated, substitute `sfPower()` instead of `sfHSD()`; use parameter values 3, 2 and 0.75 to replace the values −4,−2, and 1 in the code shown above:

```r
plot(0:100 / 100, sfPower(0.025, 0:100 / 100, 3)$spend,
  type = "l", lwd = 2,
  xlab = "Proportion of information",
  ylab = expression(paste("Cumulative \ ", alpha, "-spending")),
  main = "Kim-DeMets Spending Functions"
)
lines(
  0:100 / 100,
  sfPower(0.025, 0:100 / 100, 2)$spend,
  lty = 2,
  lwd = 2
)
lines(
  0:100 / 100,
  sfPower(0.025, 0:100 / 100, 0.75)$spend,
  lty = 3,
  lwd = 2
)
legend(
  x = c(0.0, 0.27), y = 0.025 * c(0.8, 1), lty = 1:3, lwd = 2,
  legend = c(
    expression(paste(gamma, " = 3")),
    expression(paste(gamma, " = 2")),
    expression(paste(gamma, " = 0.75"))
  )
)
```

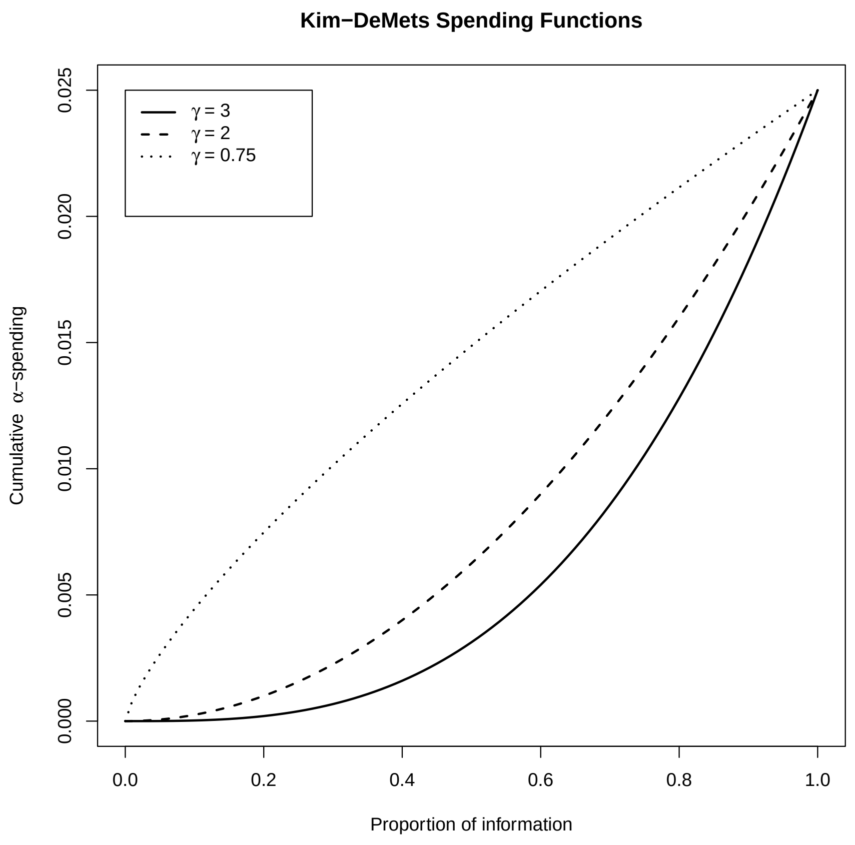


Figure 8.2

8.4 Resetting timing of analyses

When designed with a spending function, the timing and number of analyses may be altered during the course of the trial. This is very easily handled in the `gsDesign()` routine using the input arguments `n.I` and `maxn.IPlan`. We demonstrate this by example. Suppose a trial was originally designed with the call:

```
x <- gsDesign(k = 5, n.fix = 800)
x$upper$bound
#> [1] 3.252668 2.986046 2.691657 2.373666 2.025321
```

```
x$n.I
#> [1] 176.21 352.42 528.63 704.84 881.05
```

The second and third lines above show the planned upper bounds and sample sizes at analyses. Suppose that when executed the final interim was skipped, the first 2 interims were completed on time, the third interim was completed at 575 patients (instead of 529 as originally planned), the fourth interim was skipped, and the final analysis was performed after 875 patients (instead of after 882 as originally planned). The boundaries for the analyses can be obtained as follows:

```
gsDesign(
  k = 4,
  n.fix = 800,
  n.I = c(177, 353, 575, 875),
  maxn.IPlan = x$n.I[x$k]
)
#> Asymmetric two-sided group sequential design with
#> 90 % power and 2.5 % Type I Error.
#> Upper bound spending computations assume
#> trial continues if lower bound is crossed.
#>
#>                   ----Lower bounds----  ----Upper bounds-----
#>   Analysis   N    Z   Nominal p Spend+  Z   Nominal p Spend++
#>          1 177 -0.90    0.1851 0.0077 3.25    0.0006  0.0006
#>          2 353 -0.03    0.4863 0.0115 2.99    0.0014  0.0013
#>          3 575  0.90    0.8149 0.0229 2.59    0.0048  0.0040
#>          4 875  2.01    0.9779 0.0563 2.01    0.0221  0.0184
#>      Total                    0.0984                  0.0243
#> + lower bound beta spending (under H1):
#>  Hwang-Shih-DeCani spending function with gamma = -2.
#> ++ alpha spending:
#>  Hwang-Shih-DeCani spending function with gamma = -4.
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>    Theta      1      2      3      4  Total   E{N}
#>   0.0000 0.0006 0.0013 0.0040 0.0168 0.0227 480.1
#>   0.1146 0.0422 0.1683 0.3601 0.3323 0.9028 631.5
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>    Theta      1      2      3      4  Total
#>   0.0000 0.1851 0.3197 0.3219 0.1507 0.9773
#>   0.1146 0.0077 0.0115 0.0229 0.0551 0.0972
```

This design now has slightly higher power (90.4%) than the originally planned 90%. This is because the final boundary was lowered relative to the original plan when the α-spending planned for the fourth interim was saved for the final analysis by skipping the final interim. Note that all of the arguments for the original design must be supplied when the study is re-designed—in addition to adding `n.I`, which may have the same number, fewer, or more interim analyses compared to the original plan. If the sample size for the final analysis is changed, `maxn.IPlan` should be passed in as the original final sample size in order to appropriately assign α- and β-spending for the interim analyses.

8.5 Advanced spending function details

8.5.1 Spending functions as arguments

Except for the "OF", "Pocock" and "WT" examples given in Section 6.2, a spending function passed to `gsDesign()` through the arguments (upper bound) and (lower bound) must have the following calling sequence:

```
sfname(alpha, t, param)
```

where `sfname` is an arbitrary name for a spending function available from the package or written by the user. The arguments are as follows:

- `alpha`: a real value ($0 <$ `alpha` < 1). The total error spending for the boundary to be determined. This would be replaced with the following values from a call to `gsDesign()`: `alpha` for the upper bound, and either `beta` (for `test.type = 3` or `4`) or `astar` (for `test.type = 5` or `6`) for the lower bound.
- `t`: a vector of arbitrary length m of real values, $0 \leq t_1 < t_2 < \ldots t_m \leq 1$. Specifies the proportion of spending for which values of the spending function are to be computed.
- `param`: for all cases here, this is either a real value or a vector of real values. One or more parameters that (with the parameter `alpha`) fully specify the spending function. This is specified in a call to `gsDesign()` with `sfupar` for the upper bound and `sflpar` for the lower bound.

The value returned is of the class `spendfn` which was described in Chapter 8, The `spendfn` Class.

The following code and output demonstrate that the default value `sfHSD` for the upper and lower spending functions `sfu` and `sfl` is a function:

```
sfHSD
#> function (alpha, t, param)
#> {
#>     checkScalar(alpha, "numeric", c(0, Inf), c(FALSE, FALSE))
#>     checkScalar(param, "numeric", c(-40, 40))
#>     checkVector(t, "numeric", c(0, Inf), c(TRUE, FALSE))
#>     t[t > 1] <- 1
#>     x <- list(name = "Hwang-Shih-DeCani", param = param, parname = "gamma",
#>         sf = sfHSD, spend = if (param == 0) t * alpha else alpha *
#>             (1 - exp(-t * param))/(1 - exp(-param)), bound = NULL,
#>         prob = NULL)
#>     class(x) <- "spendfn"
#>     x
#> }
#> <bytecode: 0x55af7f90c6c0>
#> <environment: namespace:gsDesign>
```

Table 8.1 summarizes many spending functions available in the package. A basic description of each type of spending function is given. The table begins with standard spending functions followed by two investigational spending functions: `sfExponential()` and `sfLogistic()`. These spending functions are discussed further in Section 8.5.2 (investigational spending functions), but are included here due to their simple forms. The logistic spending function represented by `sfLogistic()` has been used in several trials. It represents a class of two-parameter spending functions that can provide flexibility not available from one-parameter families. The `sfExponential()` spending function is included here as it provides an excellent approximation of an O'Brien-Fleming design as follows:

```
gsDesign(test.type = 2, sfu = sfExponential, sfupar = 0.75)
#> Symmetric two-sided group sequential design with
#> 90 % power and 2.5 % Type I Error.
#> Spending computations assume trial stops
#> if a bound is crossed.
#>
#>            Sample
#>             Size
#>   Analysis Ratio*  Z   Nominal p  Spend
#>          1  0.338 3.51    0.0002 0.0002
#>          2  0.676 2.48    0.0066 0.0065
#>          3  1.014 2.00    0.0228 0.0183
#>      Total                       0.0250
#>
```

```
#> ++ alpha spending:
#>  Exponential spending function with nu = 0.75.
#> * Sample size ratio compared to fixed design with no interim
#>
#> Boundary crossing probabilities and expected sample size
#> assume any cross stops the trial
#>
#> Upper boundary (power or Type I Error)
#>           Analysis
#>    Theta      1      2      3 Total   E{N}
#>   0.0000 0.0002 0.0065 0.0183 0.025 1.0097
#>   3.2415 0.0519 0.5242 0.3239 0.900 0.8021
#>
#> Lower boundary (futility or Type II Error)
#>           Analysis
#>    Theta     1      2      3 Total
#>   0.0000 2e-04 0.0065 0.0183 0.025
#>   3.2415 0e+00 0.0000 0.0000 0.000
```

Table 8.1: Spending function definitions and parameterizations.

Function (parameter)	Spending function family	Functional form	Parameter (`sfupar` or `sflpar`)
`sfHSD` (`gamma`)	Hwang-Shih-DeCani	$\alpha\frac{1-\exp(-\gamma t)}{1-\exp(-\gamma)}$	Value in $[-40, 40)$. -4 = O'Brien-Fleming like; 1 = Pocock-like
`sfPower` (`rho`)	Kim-DeMets	αt^ρ	Value in $(0, +\infty)$; 3 = O'Brien-Fleming like; 1 = Pocock-like
`sfLDPocock` (none)	Pocock approximation	$\alpha\log(1+(e-1)t)$	None

Function (parameter)	Spending function family	Functional form	Parameter (`sfupar` or `sflpar`)
`sfLDOF` (none)	O'Brien-Fleming approximation	$2\times\left(1-\Phi\left(\frac{\Phi^{-1}(\alpha/2)}{\sqrt{t^{\rho}}}\right)\right)$	$\rho\in[0.05,20]$, $\rho=1$ is default
`sfLinear` (`t1`, `t2`, …, `tm`) (`p1`, `p2`, …, `pm`)	Piecewise linear specification	Specified points $0<t_1<\cdots<t_m<1$ $0<p_1<\cdots<p_m<1$	Cumulative proportion of information and error spending for given timepoints
`sfExponential` (`nu`)	Exponential	$\alpha^{t^{-\nu}}$	$(0,10]$ Recommend $\nu<1$; $0.75=$ O'Brien-Fleming-like
`sfLogistic` (`a`, `b`)	Logistic	$\alpha\frac{e^{a}(\frac{t}{1-t})^{b}}{1+e^{a}(\frac{t}{1-t})^{b}}$	$b>0$
`sfXG1` (`gamma`)	Xi-Gallo, Method 1	$2\times\left(1-\Phi\left(\frac{\Phi^{-1}(\alpha/2)-\Phi^{-1}(\gamma)\sqrt{1-t}}{\sqrt{t}}\right)\right)$	$\gamma\in[0.5,1)$, $\gamma=0.5$ same as `sfLDOF()`
`sfXG2` (`gamma`)	Xi-Gallo, Method 2	$2\times\left(1-\Phi\left(\frac{\Phi^{-1}(\alpha/2)-\Phi^{-1}(\gamma)(1-t)}{\sqrt{t}}\right)\right)$	$\gamma\in[1-\Phi(\Phi^{-1}(\alpha/2/2)),1)$
`sfXG3` (`gamma`)	Xi-Gallo, Method 3	$2\times\left(1-\Phi\left(\frac{\Phi^{-1}(\alpha/2)-\Phi^{-1}(\gamma)(1-\sqrt{t})}{\sqrt{t}}\right)\right)$	$\gamma\in(\alpha/2,1)$
`"WT"` (`Delta`)	Wang-Tsiatis bounds	$C(k,\alpha,\Delta)(i/K)^{\Delta-1/2}$	$0=$ O'Brien-Fleming bound; $0.5=$ Pocock bound
`"Pocock"` (none)	Pocock bounds		This is a special case of WT with $\Delta=1/2$

Function (parameter)	Spending function family	Functional form	Parameter (`sfupar` or `sflpar`)
`"OF"` (none)	O'Brien-Fleming bounds		This is a special case of WT with $\Delta = 0$

See also subsections below and online documentation of spending functions.

8.5.2 Investigational spending functions

When designing a clinical trial with interim analyses, the rules for stopping the trial at an interim analysis for a positive or a negative efficacy result must fit the medical, ethical, regulatory and statistical situation that is presented. Once a general strategy has been chosen, it is not unreasonable to discuss precise boundaries at each interim analysis that would be considered ethical for the purpose of continuing or stopping a trial. Given such specified boundaries, we discuss here the possibility of numerically fitting α- and β-spending functions that produce these boundaries. Commonly-used one-parameter families may not provide an adequate fit to multiple desired critical values. We present a method of deriving two-parameter families to provide some additional flexibility along with examples to demonstrate their usefulness. This method has been found to be useful in designing multiple trials, including the CAPTURE trial [The CAPTURE Investigators, 1997], the GUSTOV trial [The GUSTO V Investigators, 2001] and three ongoing trials at Merck.

One method of deriving a two-parameter spending function is to use the incomplete beta distribution which is commonly denoted by $I_x(a, b)$ where $a > 0$, $b > 0$. We let

$$\alpha(t; a, b) = \alpha I_t(a, b).$$

This spending function is implemented in `sfBetaDist()`; developing code for this is also demonstrated below in Section 8.5.4 (writing code for a new spending function).

Another approach allows fitting spending functions by solving a linear system of 2 equations in 2 unknowns. A two-parameter family of spending function is defined using an arbitrary, continuously increasing cumulative distribution function $F()$ defined on $(-\infty, \infty)$, a real-valued parameter a and a positive parameter b:

$$\alpha(t; a, b) = \alpha F(a + bF^{-1}(t)).$$

Fix two points of the spending function $0 <$ `t0` $<$ `t1` < 1 to have spending function values specified by `u0` $\times$ `alpha` and `u1` $\times$ `alpha`, respectively, where $0 <$ `u0` $<$ `u1` < 1. Equation $\alpha(t; a, b)$ now yields two linear equations with two unknowns, namely for $i = 1, 2$

$$F^{-1}(u_i) = a + bF^{-1}(t_i).$$

The four value specification of `param` for this family of spending functions is `param=c(t0, t1, u0, u1)` where the objective is that `sf(t0) = alpha*u0` and `sf(t1) = alpha*u1`. In this parameterization, all four values must be between 0 and 1 and `t0` $<$ `t1`, `u0` $<$ `u1`.

The logistic distribution has been used with this strategy to produce spending functions for ongoing trials at Merck Research Laboratories in addition to the GUSTO V trial [The GUSTO V Investigators, 2001]. The logit function is defined for $0 < u < 1$ as $\text{logit}(u) = \log(u/(1-u))$. Its inverse is defined for $x \in (-\infty, \infty)$ as $\text{logit}^{-1}(x) = e^x/(1 + e^x)$. Letting $b > 0$, $c = e^a > 0$, $F(x) = \text{logit}^{-1}(x)$ and applying $\alpha(t; a, b)$ we obtain the logistic spending function family:

$$\alpha(t; a, b) = \alpha \times \text{logit}^{-1}(\log(c) + b \times \text{logit}(u)) \tag{8.3}$$

$$= \alpha \frac{c\left(\frac{t}{1-t}\right)^b}{1 + c\left(\frac{t}{1-t}\right)^b}. \tag{8.4}$$

The logistic spending function is implemented in `sfLogistic()`. Functions are also available replacing $F()$ with the cumulative distribution functions for the standard normal distribution (`sfNormal()`), two versions of the extreme value distribution (`sfExtremeValue()` with $F(x) = \exp(-\exp(-x))$ and `sfExtremeValue2()` with $F(x) = 1 - \exp(-\exp(x))$), the central t-distribution (`sfTDist()`), and the Cauchy distribution (`sfCauchy()`). Of these, `sfNormal()` is fairly similar to `sfLogistic()`. On the other hand, `sfCauchy()` can behave quite differently. The function `sfTDist()` provides intermediary spending functions bounded by `sfNormal()` and `sfCauchy()`; it requires an additional parameter, the degrees of freedom See online help for more complete documentation, particularly for `sfTDist()`. Following is an example that plots several of these spending functions that fit through the same two points (`t1` $= 0.25$, `t2` $= 0.5$, `u1` $= 0.1$, `u2` $= 0.2$) but behave differently for $t > 1/2$.

```r
plotsf <- function(alpha, t, param) {
  plot(
    t, sfCauchy(alpha, t, param)$spend,
    lwd = 2,
    xlab = "Proportion of enrollment",
    ylab = "Cumulative spending", type = "l"
  )
  lines(
    t, sfLogistic(alpha, t, param)$spend,
    lty = 4, lwd = 2
  )
  lines(
    t, sfNormal(alpha, t, param)$spend,
    lty = 5, lwd = 2
  )
  lines(
    t, sfTDist(alpha, t, c(param, 1.5))$spend,
    lty = 2, lwd = 2
  )
  lines(
    t, sfTDist(alpha, t, c(param, 2.5))$spend,
    lty = 3, lwd = 2
  )
  legend(
    x = c(0.0, 0.3), y = alpha * c(0.7, 1), lty = 1:5, lwd = 2,
    legend = c(
      "Cauchy", "t 1.5 df", "t 2.5 df", "Logistic", "Normal"
    )
  )
}
param <- c(0.25, 0.5, 0.1, 0.2)
plotsf(0.025, t = c(0.25, 0.5, 0.75), param = param)
```

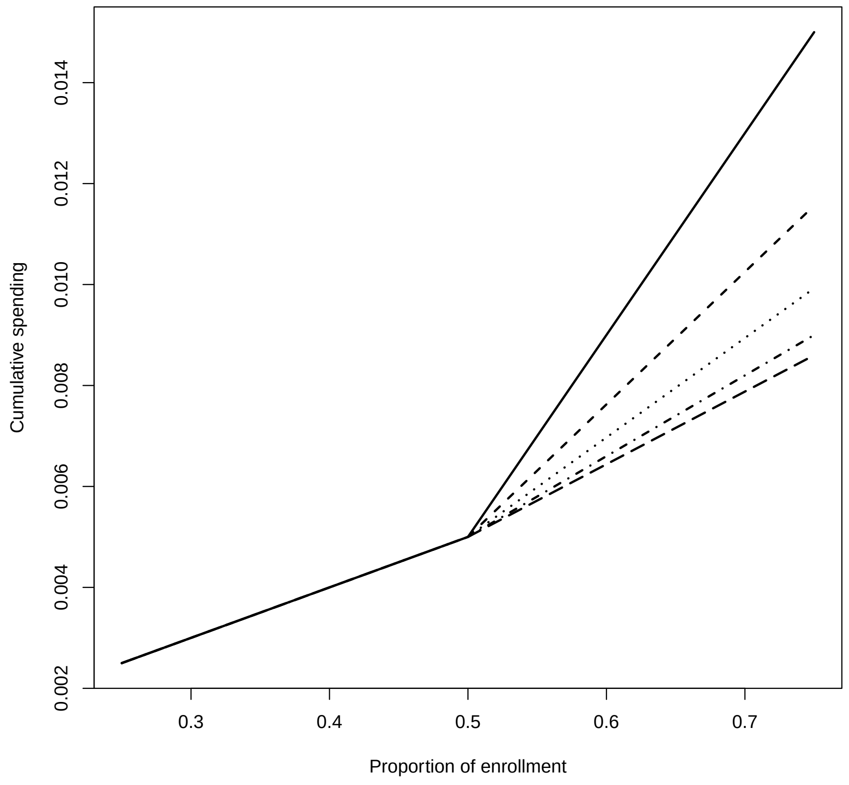


Figure 8.3

8.5.3 Optimized spending functions

The following two examples demonstrate some of the flexibility and research possibilities for the gsDesign package. The particular examples may or may not be of interest, but the strategy may be applied using a variety of optimization criteria. First, we consider finding a spending function to match a Wang-Tsiatis design. This could be useful to adjust a Wang-Tsiatis design if the timing of interim analyses are not as originally planned. Second, we replicate a result from Anderson [Anderson, 2007] which minimized expected value of the square of sample size over a family of spending functions and a prior distribution.

8.5.3.1 Approximating a Wang-Tsiatis design

We have noted several approximations of O'Brien-Fleming and Pocock spending rules using spending functions in the table above. Following is sample code to provide a good approximation of Wang-Tsiatis bounds with a given parameter Δ. This includes O'Brien-Fleming (Δ=0) and Pocock (Δ=0.5) designs. First, we define a function that computes the sum of squared deviations of the boundaries of a Wang-Tsiatis design compared to a one-parameter spending function family with a given parameter value of `x`. Note that `Delta` is the parameter for the Wang-Tsiatis design that we are trying to approximate. Other parameters are as before; recall that `test.type` should be limited to 1 or 2 for Wang-Tsiatis designs. Defaults are used for parameters for `gsDesign()` not included here.

```r
WTapprox <- function(
    x,
    alpha = 0.025,
    beta = 0.1,
    k = 3,
    sf = sfHSD,
    Delta = 0.25,
    test.type = 2) {
  # Wang-Tsiatis comparison with a one-parameter spending function
  y1 <- gsDesign(
    k = k,
    alpha = alpha,
    beta = beta,
    test.type = test.type,
    sfu = "WT",
    sfupar = Delta
  )$upper$bound
  y2 <- gsDesign(
    k = k,
    alpha = alpha,
    beta = beta,
    test.type = test.type,
    sfu = sf,
    sfupar = x
  )$upper$bound
  z <- y1 - y2
  return(sum(z * z))
}
```

We consider approximating a two-sided O'Brien-Fleming design with `alpha` = 0.025 (one-sided) using the exponential spending function. The function `nlminb()` is a standard R function used for minimization. It minimizes a function passed to it as a function of that function's first argument, which may be a vector. The first parameter of `nlminb()` is a starting value for the minimization routine. The second is the function to be minimized. The parameter `lower` below provides a lower bound for first argument to the function being passed to `nlminb()`. Following parameters are fixed parameters for the function being passed to `nlminb()`. The result suggests that for $k = 4$, an exponential spending function with $\nu = 0.75$ approximates an O'Brien-Fleming design well. Examining this same optimization for $k = 2$ to 10 suggests that $\nu = 0.75$ provides a good approximation of an O'Brien-Fleming design across this range.

```r
nu <- nlminb(
  start = 0.75,
  objective = WTapprox,
  lower = 0,
  sf = sfExponential,
  k = 4,
  Delta = 0,
  test.type = 2
)$par
nu
```

Running comparable code for `sfHSD()` and `sfPower()` illustrates that the exponential spending function can provide a better approximation of an O'Brien-Fleming design than either the Hwang-Shih-DeCani or Kim-DeMets spending functions. For Pocock designs, the Hwang-Shih-DeCani spending function family provides good approximations.

Minimizing the expected value of a power of sample size

In this example, we first define a function that computes a weighted average across a set of `theta` values of the expected value of a given power of the sample size for a design. Note that `sfupar` and `sflpar` are specified in the first input argument so that they may later be optimized using the R routine `nlminb()`. The code is compact, which is very nice for writing, but it may be difficult to interpret. A good way to see how the code works is to define values for all of the parameters and run each line from the R command prompt, examining the result.

```r
#' Expected value of the power of sample size of a trial
#' as a function of upper and lower spending parameter.
#' Get sfupar from x[1] and sflpar from x[2].
#'
#' @param val The power of the sample size for which
#'   expected values are computed.
#' @param theta A vector for which expected values
#'   are to be computed.
#' @param thetawgt A vector of the same length used to
#'   compute a weighted average of the expected values.
#'   Other parameters are as for gsDesign.
enasfpar <- function(
    x,
    timing = 1,
    theta,
    thetawgt,
    k = 4,
    test.type = 4,
    alpha = 0.025,
    beta = 0.1,
    astar = 0,
    delta = 0,
    n.fix = 1,
    sfu = sfHSD,
    sfl = sfHSD,
    val = 1,
    tol = 0.000001,
    r = 18) {
  # Derive design
  y <- gsDesign(
    k = k,
    test.type = test.type,
    alpha = alpha,
    beta = beta,
    astar = astar,
    delta = delta,
    n.fix = n.fix,
    timing = timing,
    sfu = sfu,
    sfupar = x[1],
    sfl = sfl,
    sflpar = x[2],
    tol = tol,
    r = r
  )
  # Compute boundary crossing probabilities for input theta
  y <- gsProbability(theta = theta, d = y)
  # Compute sample sizes to the val power
  n <- y$n.I^val
  # Compute expected values
  en <- n %*% (y$upper$prob + y$lower$prob)
  # Compute weighted average of expected values
  en <- sum(as.vector(en) * as.vector(thetawgt))

  return(en)
}
```

Now we use this function along with the R routine `nlminb()` which finds a minimum across possible values of `sfupar` and `sflpar`. The design derived using the code below and a more extensive discussion can be found in [Anderson, 2007]. The code above is more general than in [Anderson, 2007], however, as that paper was restricted to `test.type`=5 (the program provided there also worked for `test.type`=6).

```
# Example from Anderson (2006)
delta <- abs(qnorm(0.025) + qnorm(0.1))
# Use normal distribution to get weights
x <- normalGrid(mu = delta, sigma = delta / 2)
x <- nlminb(
  start = c(0.7, -0.8),
  enasfpar,
  theta = x$z,
  timing = 1,
  thetawgt = x$wgts,
  val = 2,
  k = 4,
  tol = 0.0000001
)
x$message
y <- gsDesign(
  k = 4,
  test.type = 5,
  sfupar = x$par[1],
  sflpar = x$par[2]
)
y
```

8.5.4 Writing code for a new spending function

Following is sample code using a cumulative distribution function for a beta-distribution as a spending function. Let B(a,b) denote the complete beta function. The beta distribution spending function is denoted for any fixed $a > 0$ and $b > 0$ by

$$\alpha(t) = \frac{\alpha}{B(a,b)} \int_0^t x^{a-1}(1-x)^{b-1} dx.$$

This spending function provides much of the flexibility of spending functions in the last subsection, but is not of the same general form. This sample code is intended to provide guidance in writing code for a new spending function, if needed.

```
# Implementation of 2-parameter version of beta distribution
# spending function.
# Assumes t and alpha are appropriately specified
# (does not check!).
sfbdist <- function(alpha, t, param) {
  # Set up return list and its class
  x <- list(
    name = "B-dist example",
    param = param,
    parname = c("a", "b"),
    sf = sfbdist,
    spend = NULL,
    bound = NULL,
    prob = NULL,
    errcode = 0,
    errmsg = "No error"
  )
  class(x) <- "spendfn"
  # Check for errors in specification of a and b
  # gsReturnError is a simple function available from the
  # package for saving errors in the returned value
  if (length(param) != 2) {
    return(gsReturnError(x,
      errcode = 0.3,
      errmsg = "b-dist example spending function parameter must be of length 2"
    ))
  }
  if (!is.numeric(param)) {
    return(gsReturnError(x,
      errcode = 0.1,
      errmsg = "Beta distribution spending function parameter must be numeric"
    ))
  }
  if (param[1] <= 0) {
    return(gsReturnError(x,
      errcode = 0.12,
      errmsg = "1st Beta distribution spending function parameter must be > 0."
    ))
  }
  if (param[2] <= 0) {
    return(gsReturnError(x,
      errcode = 0.13,
      errmsg = "2nd Beta distribution spending function parameter must be > 0."
    ))
  }
  # Set spending using cumulative beta distribution function and return
  x$spend <- alpha * pbeta(t, x$param[1], x$param[2])
  return(x)
}
```

The flexibility of this spending function is demonstrated by the following code which produces the plot below. Using `a` $= \rho$, `b` $= 1$ produces a Kim-DeMets spending function αt^ρ as shown by the solid line with ρ=2. The dashed line (`a` $= 6$, `b` $= 4$) shows a spending function that is conservative very early, but then aggressive in its spending pattern starting after about 40% of data are available. The dotted (`a` $= 0.5$, `b` $= 0.5$) and dot-dashed (`a` $= 0.6$, `b` $= 2$) show increasingly aggressive early spending. These may be useful in setting an initial high futility bound when the first part of a trial is used as a proof of concept for a clinical endpoint.

```
# Plot some beta distribution spending functions
plot(0:100 / 100, sfbdist(1, 0:100 / 100, c(2, 1))$spend,
  type = "l", lwd = 2,
  xlab = "Proportion of information",
  ylab = "Cumulative proportion of total spending",
  main = "Beta Distribution Spending Function Example"
)
lines(
  0:100 / 100, sfbdist(1, 0:100 / 100, c(6, 4))$spend,
  lty = 2, lwd = 2
)
lines(
  0:100 / 100, sfbdist(1, 0:100 / 100, c(0.5, 0.5))$spend,
  lty = 3, lwd = 2
)
lines(
  0:100 / 100, sfbdist(1, 0:100 / 100, c(0.6, 2))$spend,
  lty = 4, lwd = 2
)
legend(
  x = c(0.65, 1), y = 1 * c(0, 0.25), lty = 1:4, lwd = 2,
  legend = c(
    "a=2, b=1", "a=6, b=4", "a=0.5,b=0.5", "a=0.6, b=2"
  )
)
```

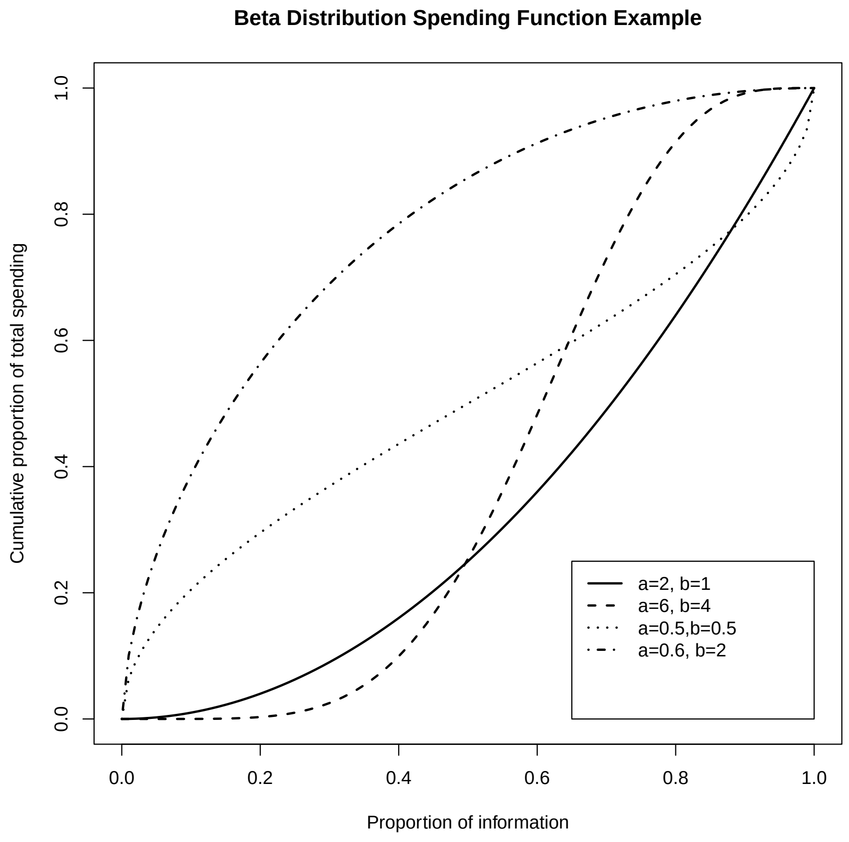


Figure 8.4

Chapter 9

Analyzing group sequential trials

We present several ways to review and interpret interim and final results from group sequential trials. Generally, regulatory agencies will have interest in well-controlled Type I error and unbiased treatment estimates.

9.1 The CAPTURE data

We consider interim [Cytel, 2007] and final [The CAPTURE Investigators, 1997] data from the CAPTURE trial. Table 9.1 provides a summary.

Table 9.1: Summary of the CAPTURE trial.

	Placebo			Experimental			
Analysis	Events	n	%	Events	n	%	Z
1	30	175	17.1%	14	175	8.0%	2.58
2	55	353	15.6%	37	347	10.7%	1.93
3	84	532	15.8%	55	518	10.6%	2.47
4	101	635	15.9%	71	630	11.3%	2.41

The interim data presented here is from finalized datasets rather than the actual data that was analyzed at the time of interim analyses. Also, we take a binomial analysis approach here using the method of Miettinen and Nurminen [Miettinen and Nurminen, 1985]; the original study analyses used the logrank test. The CAPTURE study was originally designed using symmetric bounds, and the final planned sample size was 1400 patients. The trial was stopped after analyzing the data from 1050 patients. Enrollment continued during follow-up, data entry and cleaning, adjudication and analysis of the data. By the time of the 1050 patient analysis was evaluated and the decision was made to stop the trial, a total of 1265 patients had been enrolled.

Following are the data which are then summarized including Z-values using the methods of Miettinen and Nurminen [Miettinen and Nurminen, 1985] are used, but without a continuity correction as recommended by Gordon and Watson [Gordon and Watson, 1985].

```r
library(gsDesign)

n1 <- c(175, 353, 532, 635)
n2 <- c(175, 347, 518, 630)
x1 <- c(30, 55, 84, 101)
x2 <- c(14, 37, 55, 71)
z <- testBinomial(x1 = x1, x2 = x2, n1 = n1, n2 = n2)
round(z, 2)
#> [1] 2.58 1.93 2.47 2.41
```

9.2 Testing significance of the CAPTURE data

The Z-statistics computed above can only be interpreted in context of the study design. The original design used a custom spending function which has not been published and will not be discussed further here. Also, while the trial originally had a symmetric, 2-sided design we will use a design with a futility bound here. Below we create an asymmetric design for CAPTURE with a non-binding futility rule and default upper and lower spending functions, but change the upper spending function parameter for the Hwang-Shih-DeCani spending function to $\gamma = -3$. The original plan was to perform interim analyses after 350 and 700 patients.

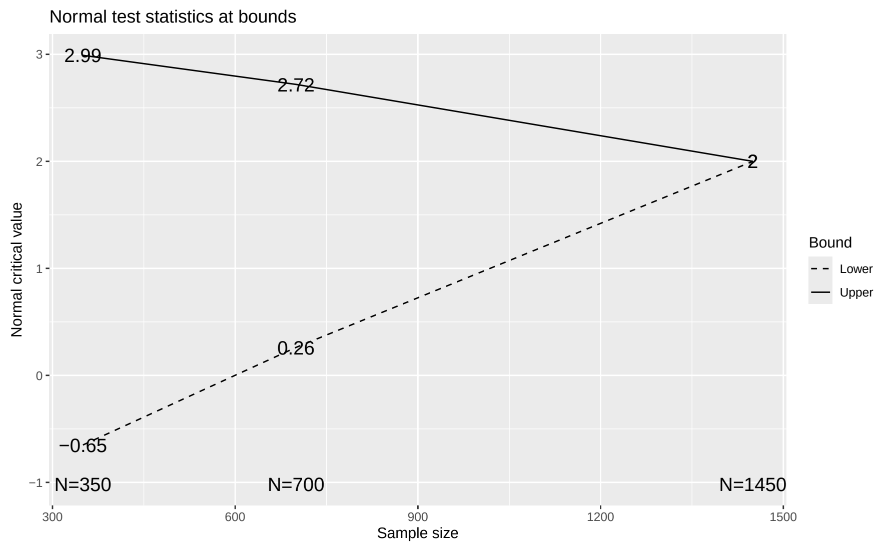
The code sequence below is as described follows:

- Compute the fixed design sample size using `nBinomial()`.
- Compute a group sequential design with standard spending functions and interim analyses after 25% and 50% of the data (first call to `gsDesign()`).
- In order to get timing of interims exactly at 350 and 700 patients, we set the timing of intervals to $350/n$, $700/n$ where n here represents the group sequential design sample size. This required 3 iterations to satisfactorily converge since the required n changes as the interim timing changes. In the final call to `gsDesign()` we include the information that the trial is designed to detect a difference in binomial rates and the presumed underlying treatment difference under the alternate hypothesis is 0.05 ($ = 0.15 - 0.10$).
- We then show the sample size and upper bounds at each analysis.
- Finally, we plot the approximate treatment differences required to cross each boundary with 3 digits of accuracy.

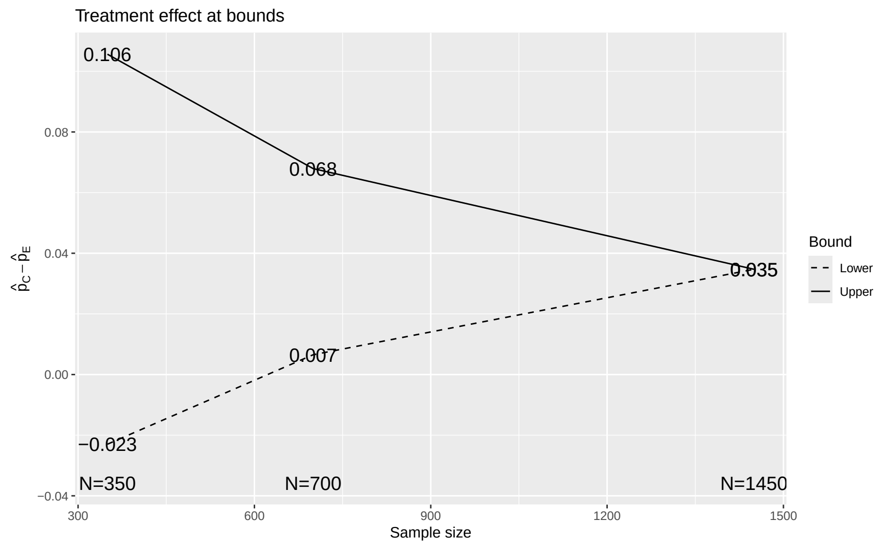
```
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
CAPTURE <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3,
  endpoint = "binomial",
  delta1 = 0.05
)
ceiling(x$n.I / 2) * 2
#> [1]  352  702 1452
```

```
x$upper$bound
#> [1] 2.990047 2.718060 1.999961
```

```
plot(CAPTURE)
```



```r
plot(CAPTURE, plottype = "xbar", dgt = c(3, 3))
```



9.3 Adding an interim analysis to a design

The original design of the CAPTURE trial had two interim analyses planned after 350 and 700 patients. As we have seen, a third interim was added at 1050 patients at the request of the data monitoring committee. The distribution theory for group sequential design requires that timing of interims is independent of interim test statistics. Thus, it would be preferable for a request for an additional interim to come from a group not familiar with the interim unblinded results; in this case, adding the requested interim was discussed with regulators who agreed to building the third interim as if there was no input based on knowledge of unblinded interim results. We will demonstrate how to appropriately control Type I error when an an analysis is added based on unblinded results when discussing conditional power and conditional error later.

The following code demonstrates how to add an interim analysis to a design when interim test statistics are not known. The arguments require are:

- The number of analyses, which is being updated to `k = 4`.
- The sample size at each analysis in `n.I`. We keep the originally planned times at 350, 700, and `CAPTURE$n.I[3]` and add an analysis after 1050 patients.
- The maximum sample size from the original plan in `maxn.IPlan`.
- The originally planned fixed design sample size in `n.fix`, original Type II error in `beta` and original spending function parameters (in this case, defaults plus `sfupar`).

```
CAPTURE3IA <- gsDesign(
  k = 4,
  n.I = c(350, 700, 1050, CAPTURE$n.I[3]),
  n.fix = n.fix,
  maxn.IPlan = CAPTURE$n.I[3],
  sfupar = -3,
  beta = 0.2
)
CAPTURE3IA$lower$bound[4]
#> [1] 1.980993
```

```
CAPTURE3IA$upper$bound[4]
#> [1] 2.039066
```

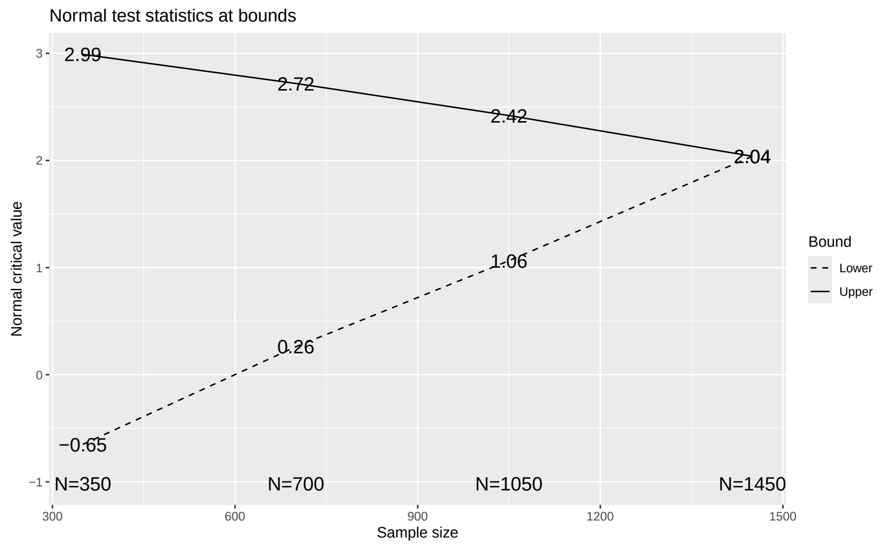
The above adjustment leaves the sample size of the two previously planned interims, the spending functions, and the final analysis sample size alone. This is done by leaving the α- and β-spending alone at these interims, which is driven by leaving the maximum sample size and spending functions alone. You can see that when a new interim analysis is added to a design without increasing the sample size, the upper and lower boundaries are not the same at the final analysis as they were previously. This is because by adding α- and β-spending at a third interim analysis, the final boundaries need to be changed to maintain the total α- and β-spending as desired. Power is measured by the probability of crossing the upper boundary under the alternate hypothesis as stored in `sum(CAPTURE3IA$upper$prob[,2])` =0.788. The fact that the upper and lower bounds are not equal to each other means the power to cross the upper bound is somewhat reduced.

To adjust the design, we change the final lower bound to be equal to the upper bound so that Type II error reflects actual decision-making; the final analysis will no longer reflect the originally planned β-spending at the final analysis

```r
cumsum(CAPTURE3IA$lower$spend)
#> [1] 0.01942596 0.05090704 0.10192428 0.20000000
```

```r
CAPTURE3IA$lower$bound[4] <- CAPTURE3IA$upper$bound[4]
CAPTURE3IA <- gsProbability(
  d = CAPTURE3IA,
  theta = CAPTURE3IA$theta
)
CAPTURE3IA$lower$prob[4, 2]
#> [1] 0.109738
```

```r
plot(CAPTURE3IA)
```



9.4 Stage-wise p-values

Fairbanks and Madsen [Madsen and Fairbanks, 1983] provide a method for computing p-values for a symmetric group sequential trial design once a boundary has been crossed. Here we will consider just p-values for positive efficacy findings for asymmetric designs. We assume for some i in $1, 2, \ldots, k$ that an upper bound is first crossed at analysis i with a test statistic values z_i. The stage-wise p-value uses the same computational method as $\alpha^+(0)$ from equation

$$\alpha^+(\theta) \equiv \sum_{i=1}^{k} \alpha_i^+(\theta)$$

$$p_{SW} = P_0\{\{Z_i \geq z_i\} \cap_{j=1}^{i-1} \{Z_j < b_j\}\}$$

This formula can still be used for the final analysis when the upper bound is never crossed. This method of computing p-values emphasizes early positive results and de-emphasizes late results. No matter how positive a result is after the first analysis, the p-value associated with a positive result will not be smaller than a first analysis result that barely crosses its bound. There is no way to compute a p-value if, for some reason, you stop a trial early without crossing a bound. For the CAPTURE data analyzed according to the default design derived above, we compute a stagewise p-value by replacing the upper boundary at the third interim analysis with the Z-value that crossed that bound. The stagewise p-value is then the probability of crossing an upper bound at the first 3 interim analyses assuming this modified bound.

```r
b <- CAPTURE3IA$upper$bound[1:2]
b <- c(b, z[3])
y <- gsProbability(
  k = 3,
  theta = 0,
  n.I = CAPTURE3IA$n.I[1:3],
  a = array(-20, 3),
  b = b
)
sum(y$upper$prob)
#> [1] 0.009259521
```

9.5 Repeated confidence intervals

Repeated confidence intervals use the nominal Type I error at each interim analysis to compute confidence bounds in the usual fashion. For the binomial analysis of the CAPTURE trial we use Miettinen and Nurminen [Miettinen and Nurminen, 1985] confidence intervals at 2 times the nominal α-level of the upper bound, $2 \times (1 - \Phi(u_k))$, $i = 1, 2, \ldots, k$.

```
gsBinomialRCI <- function(d, x1, x2, n1, n2) {
  y <- NULL
  rname <- NULL
  nanal <- length(x1)
  for (i in 1:nanal) {
    y <- c(y, ciBinomial(
      x1 = x1[i], x2 = x2[i], n1 = n1[i],
      n2 = n2[i], alpha = 2 * pnorm(-d$upper$bound[i])
    ))
    rname <- c(rname, paste("Analysis", i))
  }
  ci <- matrix(y, nrow = nanal, ncol = 2, byrow = T)
  rownames(ci) <- rname
  colnames(ci) <- c("Lower CI", "Upper CI")
  ci
}
rci <- gsBinomialRCI(
  CAPTURE3IA,
  x1[1:3],
  x2[1:3],
  n1[1:3],
  n2[1:3]
)
pdif <- (x1 / n1 - x2 / n2)[1:3]
pdif
#> [1] 0.09142857 0.04917912 0.05171713
```

```
rci
#>               Lower CI    Upper CI
#> Analysis 1 -0.01554062 0.2032692
#> Analysis 2 -0.02080474 0.1200844
#> Analysis 3 0.001147321 0.102811
```

```
plot((n1 + n2)[1:3], rci[, 2],
  ylim = c(-0.05, 0.25), xlab = "Sample size",
  ylab = expression(hat(p)[C] - hat(p)[E])
)
```

```
points((n1 + n2)[1:3], rci[, 1])
points((n1 + n2)[1:3], pdif)
```

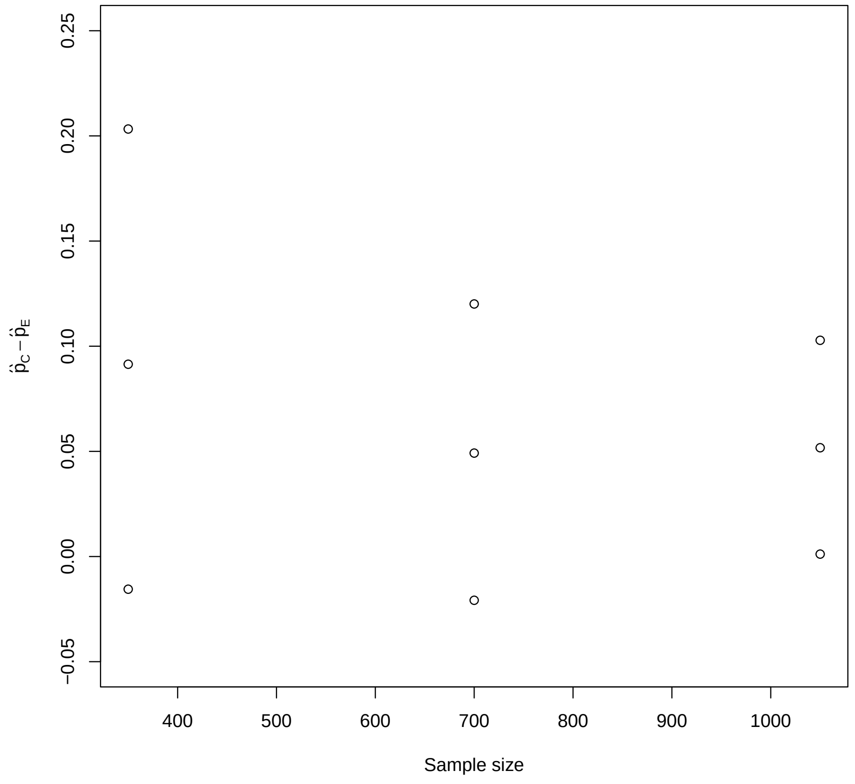


Figure 9.1

9.6 Subdensity functions

The subdensity functions defined here are useful for purposes of estimation as discussed by Jennison and Turnbull [Jennison and Turnbull, 2000], Chapter 8. In particular, these may be used for stagewise confidence intervals. We define a subdensity function for the probability that a trial continues without crossing a boundary until analysis i, and then the Z-value takes on a value z at that analysis, $i = 1, 2, \ldots, k$:

$$p_i(z|\theta) = \frac{d}{dz} P_\theta\{\{Z_i \geq z\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}, \tag{9.1}$$

These subdensity functions are used to compute $\alpha_i(\theta)$ as defined in equation

$$\alpha_i(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}, i = 1, 2, \ldots, k.$$

Normally, this will not be of great concern to the reader even though this computation will go on "behind the scenes" each time a group sequential design is derived or boundary crossing probability computed. However, a subdensity function for a design may be computed using the function `gsDensity()` as follows. In **?@fig-gsdensity**, we replicate parts of Figure 8.1 and Figure 8.2 of Jennison and Turnbull ([Jennison and Turnbull, 2000]) which show the subdensities for an O'Brien-Fleming design with four analyses.

We will later apply Equation 9.1 to compute a posterior distribution for θ given an interim test statistic $Z_i = z$ at analysis, i, $1 \leq i \leq k$.

Chapter 10

Conditional power and B-values

In some cases, rather than working with $Z_1, Z_2, \ldots, Z_k$ as in Section 2.3, it is desirable to consider variables representing incremental sets of observations between analyses. We consider a discrete version of the Brownian motion formulation of Proschan, Lan and Wittes [Proschan et al., 2006] and apply the same principles to demonstrate how to compute conditional power and conditional error using the calculation tools we have been applying for group sequential designs. Those who prefer to skip the background materials can proceed directly to Section 10.4.1.

10.1 Z-values, B-values and S-values

We let X_1, $X_2, \ldots$ be independent and identically distributed normal random variables with mean δ and variance σ^2, as before. We let $0 < n_1 < n_2 \ldots < n_k$ for some $k > 1$. For $i = 1,\, 2, \ldots, k$ define $\mathcal{I}_i = n_i/\sigma^2$ and $t_i = n_i/n_k = \mathcal{I}_i/\mathcal{I}_k$. We next define treatment effect estimates, S-values, B-values, and Z-values as follows for $i = 1, 2, \ldots, k$:

$$
\begin{aligned}
S_{i,j} &= \textstyle\sum_{m=n_i+1}^{n_j} X_m \sim N(n_{i,j}\delta, n_i\sigma^2),\\
Z_{i,j} &= S_{i,j}/(\sqrt{n_{i,j}}\sigma) \sim N(\sqrt{n_{i,j}}\theta, 1) \qquad \sim N(\sqrt{\mathcal{I}_{i,j}}\delta, 1),\\
B_{i,j} &= S_{i,j}/(\sqrt{n_k}\sigma) \quad \sim N(\sqrt{n_k}t_{i,j}\theta, t_{i,j}) \sim N(\sqrt{\mathcal{I}_k}t_{i,j}\delta, t_{i,j}).
\end{aligned}
$$

Note that $B_i = \sqrt{t_i}Z_i$, $1 \le i \le k$. Letting $\theta = \delta/\sigma$ we write a formulation of the above in terms of the standardized parameter θ:

$$\hat{\theta}_i = \hat{\delta}_i/\sigma \sim N(\theta, 1/n_i),$$

$$Z_i = \sqrt{n_i}\hat{\theta}_i \sim N(\sqrt{n_i}\theta, 1),$$

$$B_i = \sqrt{t_i}Z_i \sim N(\sqrt{n_k}t_i\theta, t_i).$$

These last forms for Z_i and B_i have been written in a general format that will apply to trials with many types of endpoints where $\hat{\delta}_i$, $\hat{\sigma}_i$ and $\mathcal{I}_i$ take different forms. Scharfstein, Tsiatis and Robbins [Scharfstein et al., 1997] have generalized work from many others by showing that if Z_i is an asymptotically efficient test for a single parameter in a parametric or semi-parametric model then the asymptotic theory holds. Proschan, Lan and Wittes [Proschan et al., 2006] provide extensive coverage of application of the Brownian motion version of B-values used here to group sequential trials with several different types of endpoints.

We now re-write bounds and boundary crossing probabilities in terms of B-values. For $i = 1, 2, \ldots, k$ we define $b_i = u_i\sqrt{t_i}$, $a_i = l_i\sqrt{t_i}$ and note that

$$\begin{aligned}\alpha_i(\theta) &= P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{l_j \leqslant Z_j < u_j\}\} \\ &= P_\theta\{\{B_i \geq b_i\} \cap_{j=1}^{i-1} \{a_j \leqslant B_j < b_j\}\}.\end{aligned} \tag{10.1}$$

Formulas for $\beta_i(\theta)$ and $\alpha_i^+(\theta)$ can be rewritten in an analogous fashion:

$$\alpha_i^+(\theta) = P_\theta\{\{B_i \geq b_i\} \cap_{j=1}^{i-1} \{B_j < b_j\}\} \tag{10.2}$$

$$\beta_i(\theta) = P_\theta\{\{B_i \leqslant a_i\} \cap_{j=1}^{i-1} \{a_j \leqslant B_j < b_j\}\} \tag{10.3}$$

10.2 Incremental formulation

We now formulate a group sequential set of outcomes in terms of independent increments to simplify conditional power calculations. Define $n_0 = S_0 = 0$ and for $0 \leq i < j \leq k$, $n_{i,j} = n_j - n_i$, $t_{i,j} = n_{i,j}/n_k$, $\mathcal{I}_{i,j} = n_{i,j}/\sigma^2$

$$S_{i,j} = \sum_{m=n_i+1}^{n_j} X_m \sim N(n_{i,j}\delta, n_i\sigma^2),$$

$$Z_{i,j} = S_{i,j}/(\sqrt{n_{i,j}}\sigma) \sim N(\sqrt{n_{i,j}}\theta, 1) \quad \sim N(\sqrt{\mathcal{I}_{i,j}}\delta, 1),$$

$$B_{i,j} = S_{i,j}/(\sqrt{n_k}\sigma) \sim N(\sqrt{n_k}t_{i,j}\theta, t_{i,j}) \sim N(\sqrt{\mathcal{I}_k}t_{i,j}\delta, t_{i,j}).$$

Analogous to before, $B_{i,j} = \sqrt{t_{i,j}} Z_{i,j}$ and we can apply the ultimate conditional power and conditional error calculations below broadly to group sequential designs for which the general asymptotic theory is applicable. For $0 \leq i < j \leq k$ we note that $B_i = B_{0,i}$ and that $B_j = B_i + B_{i,j}$ where B_i and $B_{i,j}$ are independent and normally distributed. Thus, the conditional distribution of B_j assuming $B_i = c$ for some $1 < i < j \leqslant k$ and constant c is normal with mean $c + \sqrt{n_k} t_{i,j}\theta$ and variance $t_{i,j}$.

10.3 Simple conditional power and conditional error

Proschan, Lan and Wittes [Proschan et al., 2006] and others compute a simple version of conditional boundary crossing probabilities that ignores interim stopping boundaries between an interim analysis that we condition on and some future analysis of interest. That is, assuming $1 \leq i < j \leq k$ we assume $B_i = c$ and we compute conditional power as

$$\begin{aligned} P\{B_j \geq b_j | B_i = c\} &= P\{B_{i,j} \geq b_j - c\} \\ &= 1 - \Phi\left(\frac{b_j - c - \theta t_{i,j}\sqrt{n_k}}{\sqrt{t_{i,j}}}\right) \\ &= 1 - \Phi\left(\frac{b_j - c - \delta t_{i,j}\sqrt{\mathcal{I}_k}}{\sqrt{t_{i,j}}}\right) \end{aligned} \tag{10.4}$$

10.3.1 Simple conditional power for the CAPTURE trial

We consider the interim data as previously displayed in Chapter 9, but restrict ourselves initially to the 2 interim analysis design. Recall that test statistics are in `z` and the design in `CAPTURE`. We apply Equation 10.4 to compute the simple conditional power and conditional error of crossing each future bound at the time of the first interim analysis. For each analysis, we place the information fractions (fractions of final sample size in this case) in `ti` and the B-value bounds for the design in `b`. We use the standardized parameter θ formulation from equation Equation 10.4. The observed B-values corresponding to the first 3 analyses are placed in `bobs`. We then store the effect size powered for in `theta1` and the observed interim effect size at the first analysis estimated by $\hat{\theta}_1 = Z_1/\sqrt{n_1}$ in `thetahat`. Finally, we compute the standard deviation for $B_{1,j}$ for $j = 2, 3, 4$ as $t_j - t_1$ and store it in `sd1j`.

```r
library(gsDesign)

n1 <- c(175, 353, 532, 635)
```

```r
n2 <- c(175, 347, 518, 630)
x1 <- c(30, 55, 84, 101)
x2 <- c(14, 37, 55, 71)
z <- testBinomial(x1 = x1, x2 = x2, n1 = n1, n2 = n2)

n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
CAPTURE <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3,
  endpoint = "binomial",
  delta1 = 0.05
)
```

```r
n <- (n1 + n2)[1:2]
ti <- c(n / CAPTURE$n.I[3], 1)
b <- CAPTURE$upper$bound * sqrt(ti)
bobs <- z[1:2] * sqrt(ti[1:2])
theta1 <- CAPTURE$delta
thetahat <- z[1:2] / sqrt(n)
t1j <- ti[2:3] - ti[1]
n1j <- c(n[2], CAPTURE$n.I[3]) - n[1]
```

```r
sd1j <- sqrt(t1j)
b1j <- b[2:3] - bobs[1]
```

The simple conditional error after the first interim analysis is

```r
simpCError <- pnorm(
  b1j,
  sd = sd1j,
  mean = 0,
  lower = FALSE
)
simpCError
#> [1] 0.1028575 0.2001852
```

Plugging the originally powered effect size produces one conditional power estimate.

```r
simpCPwr1 <- pnorm(
  b1j,
  sd = sd1j,
  mean = theta1 * sqrt(CAPTURE$n.I[3]) * t1j,
  lower = FALSE
)
simpCPwr1
#> [1] 0.5596153 0.9523688
```

The conditional power assuming the observed interim effect size is

```r
simpCPwrHat <- pnorm(
  b1j,
  sd = sd1j,
  mean = thetahat * sqrt(CAPTURE$n.I[3]) * t1j,
  lower = FALSE
)
simpCPwrHat
#> [1] 0.9056190 0.9421038
```

10.4 Conditional power and conditional error

As an alternative to futility bounds based on β-spending, stopping rules for futility are sometimes based on the conditional power of a positive trial given the value of a test statistic at an interim analysis. Thus, we consider the conditional probabilities of boundary crossing for a group sequential design given an interim result. Assume $1 \leq i < j \leqslant k$ and $B_i = c$. The conditional probabilities for first crossing an upper boundary at analysis j given $B_i = c$ is

$$\begin{aligned}\alpha_{i,j}(\theta|B_i = c) = & \quad P_\theta\{\{B_j \geqslant b_j\} \cap_{m=i+1}^{j-1} \{a_m \leqslant B_{0,m} < b_m\}|B_{0i} = c\} \\ = & \quad P_\theta\{\{B_{ij} \geqslant b_j - c\} \cap_{m=i+1}^{j-1} \{a_m - c \leqslant B_{im} < b_m - c\}\}\end{aligned} \tag{10.5}$$

Formulas for $\alpha^+_{i,j}(\theta|B_i = c)$ and $\beta_{i,j}(\theta|c)$ can be written analogously:

$$\alpha^+_{i,j}(\theta|B_i = c) = \quad P_\theta\{\{B_{i,j} \geqslant b_j - c\} \cap_{m=i+1}^{j-1} \{B_{im} < b_m - c\}\} \tag{10.6}$$

$$\beta_{i,j}(\theta|B_i = c) = \quad P_\theta\{\{B_{i,j} \leqslant b_j - c\} \cap_{m=i+1}^{j-1} \{a_m - c \leqslant B_{i,m} < b_m - c\}\} \tag{10.7}$$

Given the above and Equation 10.1, conditional boundary crossing probabilities can be computed using the same numerical tools as unconditional boundary crossing probabilities. The conditional design generated is that of Muller and Schafer [Müller and Schäfer, 2001].

Given the above and Equation 10.1, conditional boundary crossing probabilities can be computed using the same numerical tools as unconditional boundary crossing probabilities. We rewrite the incremental B-value bounds as incremental Z-value bounds as these are values produced in the function `gsCP()` provided below. Recalling that the incremental Z-value is denoted $Z_{i,j} = B_{i,j}/\sqrt{t_{i,j}}$ for $0 \leq i < j \leq k$. If the interim B-value at analysis i is c as above, we define $z_i = c/\sqrt{t_i}$. Then, crossing an upper bound with $B_{i,j} \geq b_j - c$ later at analysis $j > i$ is equivalent to

$$Z_{i,j} = \frac{B_{i,j}}{\sqrt{t_{i,j}}} \geq \frac{b_j - c}{\sqrt{t_{i,j}}} = \frac{u_j\sqrt{t_j} - z_i\sqrt{t_i}}{\sqrt{t_{i,j}}}.$$

The incremental Z-value upper and lower bounds are defined respectively by

$$u_{i,j}(z_i) = \frac{u_j\sqrt{t_j} - z_i\sqrt{t_i}}{\sqrt{t_{i,j}}} \tag{10.8}$$

and

$$l_{i,j}(z_i) = \frac{l_j\sqrt{t_j} - z_i\sqrt{t_i}}{\sqrt{t_{i,j}}}. \tag{10.9}$$

10.4.1 Application to the CAPTURE trial

We now examine the observations after the initial interim that would be required to cross the bounds for the original design. The function `gsCP()` computes a new set of boundaries $l_{i,j}$ and $u_{i,j}$ from Equation 10.8 and Equation 10.9 based on results at interim i. Values of θ for which conditional power is computed are, by default,

- $\theta = 0$, for conditional error.
- $\theta = \hat{\theta}$ to compute conditional power at the interim trend as in, for example, Bauer and Köhne [Bauer and Kohne, 1994], Proschan and Hunsberger [Proschan and Hunsberger, 1995] and Cui, Hung and Wang [Cui et al., 1999].
- $\theta = \delta$, the treatment effect at which the trial was originally powered, as in Liu and Chi [Liu and Chi, 2001].

```r
xCP1 <- gsCP(x = CAPTURE, i = 1, zi = z[1])
```

We compute the conditional power after the first interim analysis based on the interim estimate of θ,

```r
sum(xCP1$upper$prob[, 1])
#> [1] 0.9999172
```

the conditional error when $\theta = 0$,

```r
sum(xCP1$upper$prob[, 2])
#> [1] 0.2449957
```

and the conditional power based on $\theta = \delta$ where δ is the value for which the trial was originally powered.

```r
sum(xCP1$upper$prob[, 3])
#> [1] 0.9577563
```

The probability of crossing at the second interim analysis given the first is the same as previously computed in Section 10.3, interim treatment effect size, $\theta = \theta_1$, the effect size originally powered for, and $\theta = 0$, respectively.

```r
xCP1$upper$prob[1, 1]
#> [1] 0.905619
```

```r
simpCPwrHat[1]
#> [1] 0.905619
```

```
xCP1$upper$prob[1, 3]
#> [1] 0.5595968
```

```
simpCPwr1[1]
#> [1] 0.5596153
```

```
xCP1$upper$prob[1, 2]
#> [1] 0.1028575
```

```
simpCError[1]
#> [1] 0.1028575
```

As you can see, the conditional probability of success varies considerably depending on the value of θ. We will see later that there is substantial uncertainty concerning θ at the first interim, thus calling into question use of any particular θ value to compute conditional power.

Chapter 11

Bayesian design properties

All of the properties of group sequential designs we have considered so far have depended on knowing an exact value θ measuring treatment effect. Some evaluations may benefit from taking into account uncertainty concerning θ. Rather than assuming a fixed value of θ to compute quantities such as power, Type I error and expected sample size, we consider these quantities based on a prior distribution for θ. Power and Type I error are replaced by probability of success when averaging over a prior distribution. By assigning a value to a trial outcome, the prior distribution for θ can be used to compute the value of a trial design. Given built-in optimization functions in R, it is then easy to apply gsDesign functions to optimize specific characteristics of a trial design such as a futility bound or spending function.

After an interim result, we can again update the uncertainty about θ by computing a posterior distribution. Conditional power can be replaced by predictive power by averaging conditional power over this posterior distribution. The value of the trial can be replaced by the predictive value which conditions on an interim outcome. We can also compute prediction intervals for future trial results conditioned on an interim outcome. All of these measures can be based on a specific interim result or (blinded) knowledge that no trial boundary has been crossed. The latter may be particularly useful for a sponsor to update their internal estimates for the ultimate probability that a trial is successful.

In summary, the following questions may be answered using a Bayesian approach:

- What is the probability of success of the trial?
- Based on a specified utility function, what is the value of a trial design?
- What is the posterior for θ given an interim result?
- What is the predictive distribution of future observations given an interim result?
  - What is the predictive probability of success?

- Can we compute prediction intervals for future results?
- What is the value of a trial conditional on the interim result?

Examples in this section compute answers to all of these questions when framed in terms of a particular trial.

11.1 Normal densities

The primary tool for computing group sequential probabilities is application of numerical integration of a normal density function. The `normalGrid()` function is a lower-level function used by `gsProbability()` and `gsDesign()` that is normally obscured from the user. For Bayesian computations with a normal prior distribution, such as here, it can be quite useful. Since this also has many potential applications in areas - not restricted to group sequential design - we have made this function available to users. The reason for making a separate function rather than just using the R function `pnorm()` is that `normalGrid()` computes grid weights for numerical integration using Simpon's method as recommended by Jennison and Turnbull in Chapter 19 [Jennison and Turnbull, 2000]. Figure 11.1 shows a plot of weights for numerical integration from a standard normal distribution as generated below. We can integrate out the expected value of a function of a standard normal distribution by multiplying the value of the function at each point on the grid by the grid weights and summing to integrate. This is illustrated by computing the $E\{Z^2\} = 1$ below. The density evaluated at values in `g$z` are stored in `g$density`, if needed.

```r
library(gsDesign)

g <- normalGrid(mu = 0, sigma = 1)
sum(g$z^2 * g$wgts)
#> [1] 1.000001
```

```r
plot(g$z, g$wgts)
```

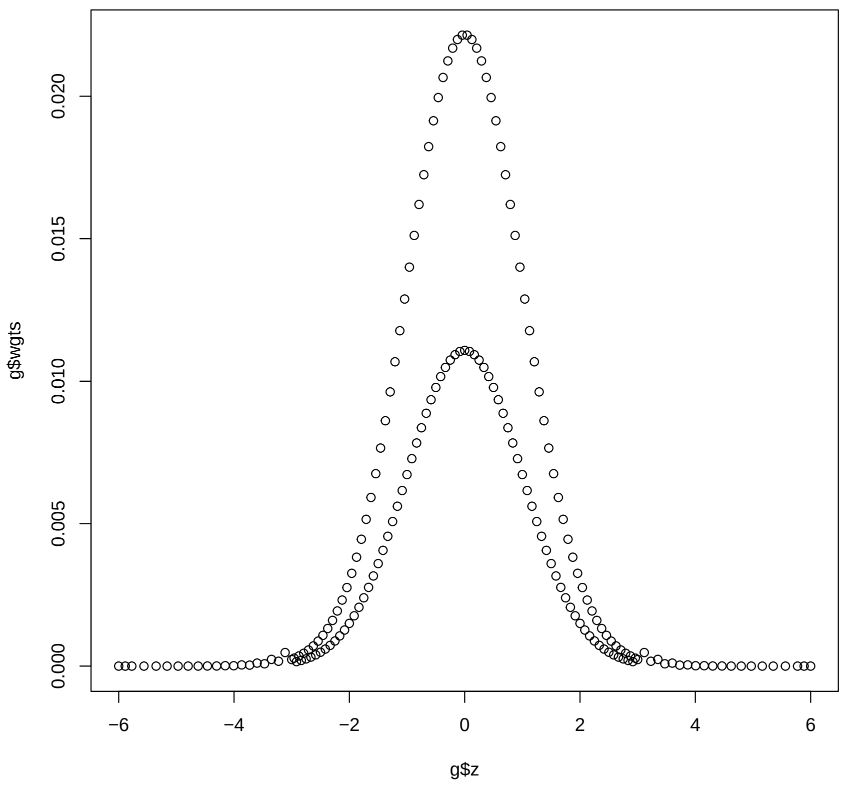


Figure 11.1: Weights for numerical integration from a standard normal distribution.

We demonstrate the use of **`normalGrid()`** to generate a prior distribution for the parameter of interest in a group sequential design below. This will then be used in various Bayesian applications in later sections. Here we simply describe the use of the **`normalGrid()`** function used to work with normal densities.

Next we plot normal densities with mean 0 and 0.5 and standard deviations 1 and 2, respectively, in Figure 11.2 and Figure 11.3.

```
g <- normalGrid(mu = 0, sigma = 1)
plot(g$z, g$wgts, main = "mean=0, standard deviations=1")
```

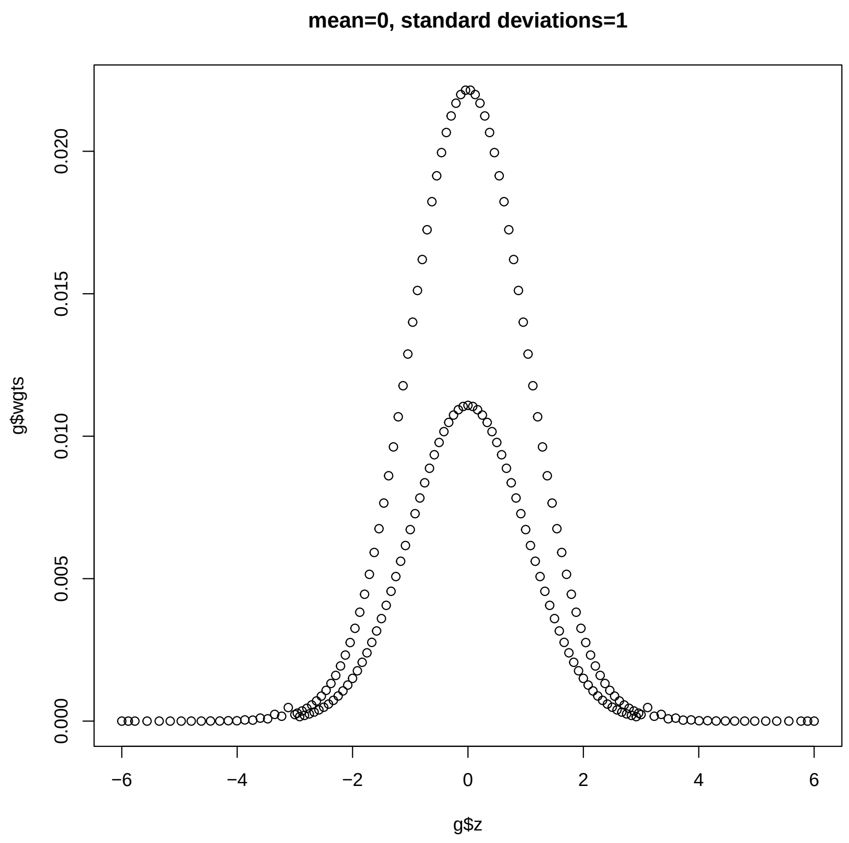


Figure 11.2

```r
g <- normalGrid(mu = 0.5, sigma = 2)
plot(g$z, g$wgts, main = "mean=0.5, standard deviations=2")
```

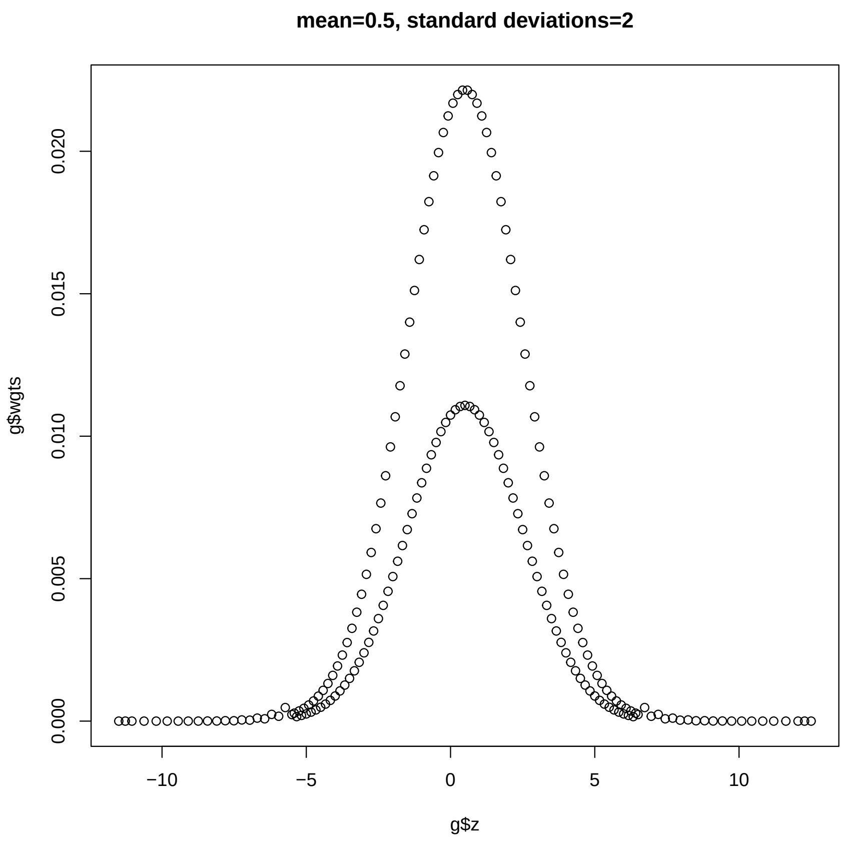


Figure 11.3

We then add a normal density with range -5 to -2 with mean -4 and standard deviation 1.5 in Figure 11.4.

```
d1 <- normalGrid()
d2 <- normalGrid(mu = 0.5, sigma = 2)
minx <- min(d1$z, d2$z)
maxx <- max(d1$z, d2$z)
plot(
  x = d2$z,
  y = d2$density,
  type = "l",
  xlim = c(minx, maxx),
  ylim = c(0, max(d1$density)),
  xlab = "z",
```

```r
  ylab = "Density",
  lwd = 2,
  main = "Normal density examples."
)
lines(x = d1$z, y = d1$density, lty = 2, lwd = 2)
d3 <- normalGrid(mu = -4, sigma = 1.5, bounds = c(-5, -2))
lines(x = d3$z, y = d3$density, lwd = 2, lty = 3)
legend(
  x = c(4, 7),
  y = c(0.27, 0.4),
  lty = c(2, 1, 3),
  lwd = 2,
  legend = c("d1", "d2", "d3")
)
```

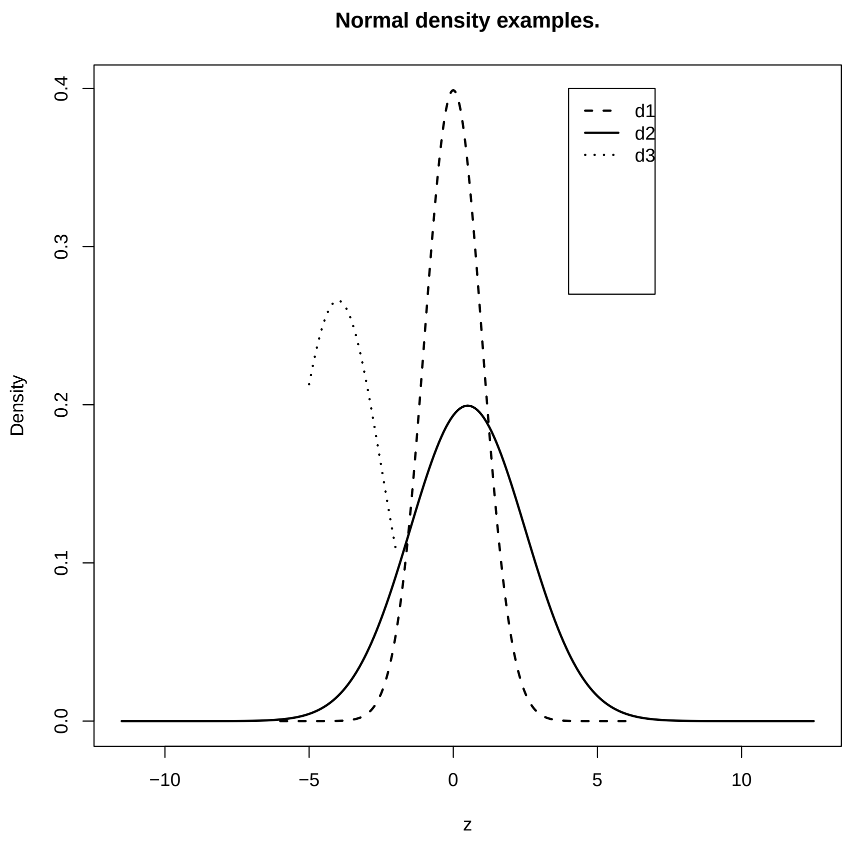


Figure 11.4

11.2 The posterior distribution for θ

For a given parameter sample space of real-values Θ, denote the prior distribution function for $\theta \in \Theta$ at the beginning of the trial by $G(\theta)$. This may represent a continuous or discrete distribution or a combination of the two. The joint sub-density of θ, the interim test statistics not crossing a bound prior to analysis i, and an interim value z at analysis i, $1 \le i \le k$ is

$$p_i(z|\theta)\frac{d}{d\theta}G(\theta).$$

To obtain a posterior distribution, we integrate over all possible values of $\theta \in \Theta$ to get a marginal density for $Z_i = z$ at analysis i for the denominator in the following equation for the posterior distribution of θ given no boundary crossing prior to analysis i and the interim test statistic $Z_i = z$, $1 \leq i \leq k$:

$$dG_i(\theta|z) = \frac{p_i(z|\theta)dG(\theta)/d\theta}{\int_{\eta\in\Theta} p_i(z|\eta)dG(\eta)/d\eta G_i(z,\eta)}. \tag{11.1}$$

When a sufficient statistic $\hat{\theta}$ is available to estimate θ at an interim analysis, the marginal distribution can be factored into the joint distribution of the sufficient statistic $\hat{\theta}$ and θ times a function of the individual data points that is independent of θ. This means computing the posterior distribution can ignore the individual data points. Proschan, Lan and Wittes [Proschan et al., 2006] show the posterior distribution under the assumption of a normal prior. Their formulation is that $\theta = E\{Z_k\}$, where Z_k is the test statistic at the final planned analysis. The formulation here has $\sqrt{n_k}\theta = E\{Z_k\}$, and we translate the Proschan, Lan and Wittes formulation appropriately. Assuming the prior distribution

$$\theta \sim N(\theta_0, \sigma_0^2)$$

the posterior distribution conditioning on an interim test statistic z_i (or $B_i = b = z_i\sqrt{t_i}$) at interim i for $1 \leq i \leq k$ is normal with mean

$$E\{\theta|Z_i = z_i\} = \frac{\theta_0/\sigma_0^2 + \sqrt{n_i}z_i}{1/\sigma_0^2 + n_i} = \frac{\theta_0/\sigma_0^2 + \sqrt{n_k}b}{1/\sigma_0^2 + n_i} \tag{11.2}$$

and variance

$$\text{Var}\{\theta|Z_i = z_i\} = (1/\sigma_0^2 + n_i)^{-1}. \tag{11.3}$$

Note that the conditional variance does not depend on z_i. The value $E\{\theta|Z_i = z_i\}$ is a weighted average of the prior mean and an estimate of θ based on data. The weights are the statistical information (inverse of the variance) for each estimate. The total statistical information for the combined estimate (inverse of the posterior variance) is the sum of the statistical information for each estimate.

To demonstrate the above calculation, we again consider the CAPTURE study and assume a normal prior distribution for θ with a mean of $.4\theta_1$ where θ_1 is the alternate hypothesis value of θ and an uninformative standard deviation of $6\theta_1$. Code to generate this prior using `normalGrid()` is as follows:

```r
n.fix <- nBinomial(p1 = 0.15, p2 = 0.1, beta = 0.2)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
x <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3
)
CAPTURE <- gsDesign(
  k = 3,
  n.fix = n.fix,
  timing = c(350, 700) / x$n.I[3],
  beta = 0.2,
  sfupar = -3,
  endpoint = "binomial",
  delta1 = 0.05
)
```

```r
CAPTURE$delta
#> [1] 0.07565793
```

```r
sigma1sq <- 6 * CAPTURE$delta^2
mu <- 0.4 * CAPTURE$delta
prior <- normalGrid(mu = mu, sigma = sqrt(sigma1sq))
```

Next we compute the posterior distribution and, following each analysis, the posterior density. We then see the posterior probability that $\theta \leq 0$ a priori and after each analysis and compare this to the nominal p-value for each z-statistic. The prior we have used is relatively uninformative, so the p-values and posterior probabilities are quite similar. The slight favorable prior lowers the posterior probability compared to nominal p-values. The interim data and statistics are as we have computed them in Chapter 9.

```r
n1 <- c(175, 353, 532, 635)
n2 <- c(175, 347, 518, 630)
x1 <- c(30, 55, 84, 101)
x2 <- c(14, 37, 55, 71)
z <- testBinomial(x1 = x1, x2 = x2, n1 = n1, n2 = n2)
```

```r
n <- n1 + n2
postmean <- (mu / sigma1sq + z * sqrt(n)) / (1 / sigma1sq + n)
postvar <- 1 / (1 / sigma1sq + n)
round(postmean, 3)
#> [1] 0.130 0.071 0.075 0.067
```

```r
round(sqrt(postvar), 3)
#> [1] 0.051 0.037 0.030 0.028
```

```r
round(pnorm(-c(mu, postmean) / sqrt(c(sigma1sq, postvar))), 4)
#> [1] 0.4351 0.0058 0.0275 0.0068 0.0081
```

```r
round(1 - pnorm(z), 4)
#> [1] 0.0049 0.0271 0.0067 0.0081
```

Next we compute the posterior density for θ at each analysis using Equation 11.2 and Equation 11.3.

```r
post1den <- normalGrid(mu = postmean[1], sigma = sqrt(postvar[1]))
post2den <- normalGrid(mu = postmean[2], sigma = sqrt(postvar[2]))
post3den <- normalGrid(mu = postmean[3], sigma = sqrt(postvar[3]))
post4den <- normalGrid(mu = postmean[4], sigma = sqrt(postvar[4]))
```

In order to do a check, we could have also computed the posterior after the first interim analysis with

```r
post2 <- gsPosterior(x = CAPTURE, i = 2, zi = z[2], prior = prior)
```

If z represents a set of values (e.g., an interval) and $p_i(z|\theta)$ the probability that $Z_i \in z$ given θ, then we can still apply the equation posterior. The likelihood is computed for not crossing a bound prior to analysis `i` and being in the interval specified in the 2-values specified in `zi`. The default if `zi` is not given is to use the interval between the upper an lower bound at the analysis specified in `i`, as in the following example. We compute such posterior distributions for the first 3 analyses.

```r
CAPTURE3IA <- gsDesign(
  k = 4,
  n.I = c(350, 700, 1050, CAPTURE$n.I[3]),
  n.fix = n.fix,
  maxn.IPlan = CAPTURE$n.I[3],
  sfupar = -3,
  beta = 0.2
)
```

```r
post1b <- gsPosterior(x = CAPTURE3IA, i = 1, prior = prior)
post2b <- gsPosterior(x = CAPTURE3IA, i = 2, prior = prior)
post3b <- gsPosterior(x = CAPTURE3IA, i = 3, prior = prior)
```

Finally, we plot the prior and posteriors where the interim statistics are known in black as well as the posteriors when it is only known that a boundary has not been crossed in red. Since there is less information in the latter, red posteriors, they are more disperse than the corresponding posterior when the interim test statistic is known. The central location for the latter posteriors is also closer to the original prior than the posteriors based on exact results where the posteriors more shift to match the data.

```r
plot(
  post4den$z, post4den$density,
  type = "l", lty = 5,
  xlab = expression(theta), ylab = "Density"
)
lines(post3den$z, post3den$density, lty = 4)
lines(post2den$z, post2den$density, lty = 3)
lines(post1den$z, post1den$density, lty = 2)
lines(prior$z, prior$density)
legend(
  x = c(-0.1, -0.03), y = c(6, 14), lty = c(1:5), lwd = 2,
  legend = c(
    "Prior", "Analysis 1", "Analysis 2", "Analysis 3", "Analysis 4"
  )
)
lines(post1b$z, post1b$density, lty = 2, col = 2)
lines(post2b$z, post2b$density, lty = 3, col = 2)
lines(post3b$z, post3b$density, lty = 4, col = 2)
```

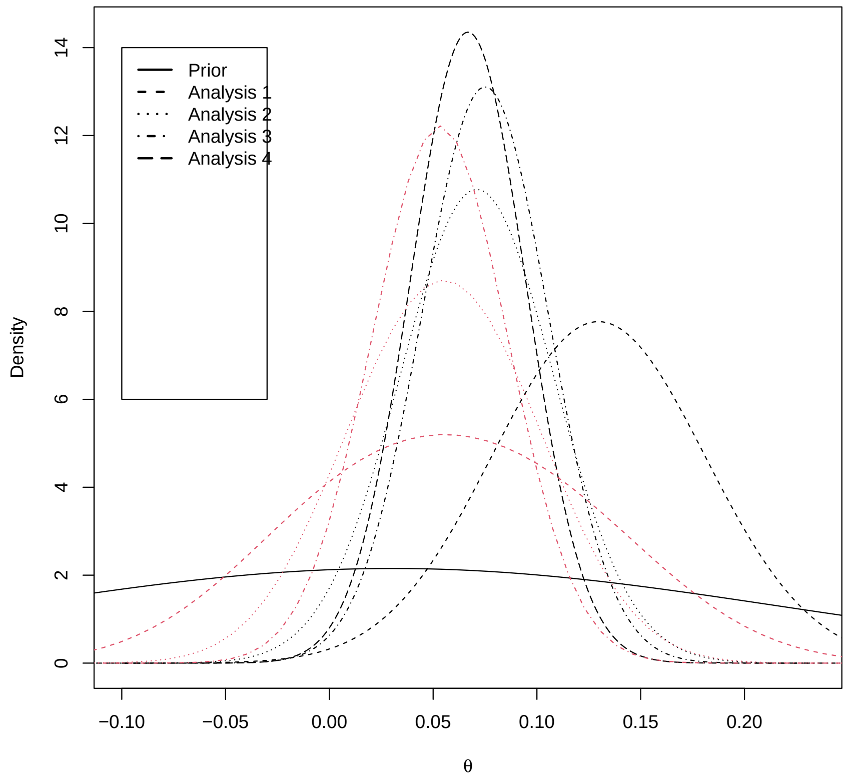


Figure 11.5

11.3 Bayes credible intervals

A Bayes credible interval can be computed using the posterior distribution for θ. Let $G_i^{-1}(\cdot|Z_i = z_i)$ be the inverse of $G_i(\cdot|Z_i = z_i)$. For an interval at level $1 - \alpha/2$, the Bayes credible interval for θ given $Z_i = z_i$ is $(G_i^{-1}(\alpha/2|Z_i = z_i), G_i^{-1}(1 - \alpha/2|Z_i = z_i)$. We continue the example with a normal prior and posterior. The Bayes credible interval is simply quantiles from a normal distribution with parameters Equation 11.2 and Equation 11.3. Extending our previous example, this can be computed for each analysis of CAPTURE using the inverse standard normal function `qnorm()` as follows

```
lCredInt <- postmean + qnorm(0.025) * sqrt(postvar)
uCredInt <- postmean + qnorm(0.975) * sqrt(postvar)
```

```
CredInt <- as.matrix(cbind(lCredInt, postmean, uCredInt))
rownames(CredInt) <- paste("Analysis", 1:4)
colnames(CredInt) <- c("Lower", "Mean", "Upper")
CredInt
#>                   Lower       Mean     Upper
#> Analysis 1  0.028963082 0.12962419 0.2302853
#> Analysis 2 -0.001507327 0.07107813 0.1436636
#> Analysis 3  0.015387474 0.07505169 0.1347159
#> Analysis 4  0.012299533 0.06678258 0.1212656
```

11.4 Predictive power

The credible intervals for θ are fairly wide, suggesting that computing conditional power based on a fixed value of θ may be misleading. Predictive power extends the concept of conditional power by incorporating uncertainty about the parameter θ into computations. Recall that we have denoted the conditional probability of crossing an efficacy boundary at analysis j given an interim test statistic z_i at analysis $i < j$ and parameter θ as $\alpha_{i,j}(\theta|z_i)$. Given a posterior distribution $G_i(\theta|z_i)$ for θ after observing an interim test statistic z_i defined on some set of values Θ, the predictive probability of crossing the efficacy boundary at analysis $j > i$ is defined in Equation 11.4.

$$\alpha_{i,j}(z_i, G_i) = \int_\Theta \alpha_{i,j}(\theta|z_i) dG_i(\theta|z_i). \tag{11.4}$$

Predictive probability is computed using the function `gsPP()` as demonstrated below. The first line computes the predictive power to cross a bound at any future analysis. The second line, by specifying `total=FALSE`, computes the predictive power of crossing a bound at the individual future planned analyses.

```
gsPP(
  x = CAPTURE,
  zi = z[1],
  i = 1,
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 0.9633891
```

```
gsPP(
  x = CAPTURE,
  zi = z[1],
  i = 1,
  theta = prior$z,
  wgts = prior$wgts,
  total = FALSE
)
#>            [,1]
#> [1,] 0.7984714
#> [2,] 0.1649177
```

11.5 Predictive distribution for the normal case

For $1 \leq i < j \leq K$, the simple conditional distribution for B_j given $B_i = b$ is easily extended to a predictive distribution when the prior for θ is normally distributed. Given $B_i = b$, we consider that

$$B_j = b + B_{i,j} = b + (t_j - t_i)\sqrt{n_k}\theta + (B_{i,j} - (t_j - t_i)\sqrt{n_k}\theta).$$

Conditioning on θ, $B_{i,j} - (t_j - t_i)\sqrt{n_k}\theta \sim N(0, t_j - t_i)$ and because this does not depend on θ it has the same unconditional distribution. We know the posterior distribution for θ given $B_i = b$ is normal with mean from Equation 11.2 and variance from Equation 11.3. Thus, we derive that the predictive distribution for B_j given $B_i = b$ is normal with

$$E\{B_j|B_i = b\} = b + E\{B_{i,j}\} = b + \sqrt{n_k}(t_j - t_i)E\{\theta|B_i = b\} \qquad (11.5)$$

and

$$\text{Var}\{B_j|B_i = b\} = (t_j - t_i)\left(1 + \frac{n_k(t_j - t_i)}{1/\sigma_0^2 + n_i}\right) \qquad (11.6)$$

We now the compute predictive probability of a positive trial at the second interim analysis of the CAPTURE trial using this posterior distribution and compare to `gsPP()`. Recall that we have already computed the posterior mean and variance for θ at each analysis in `postmean` and `postvar`, respectively, and the proportion of final planned statistical information at the planned analyses in `ti`.

```
gsPP(
  x = CAPTURE,
  zi = z[2],
  i = 2,
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 0.7643489
```

```
ti <- c((n1 + n2)[1:2] / CAPTURE$n.I[3], 1)
```

```
b <- z[2] * sqrt(ti[2])
pimean <- b + postmean[2] * (ti[3] - ti[2]) * sqrt(CAPTURE$n.I[3])
pivar <- (postvar[2] + 1) * (ti[3] - ti[2])
pivar <- (postvar[2] * (CAPTURE$n.I[3] - 700) + 1) * (ti[3] - ti[2])
bfinal <- CAPTURE$upper$bound[3]
pnorm(bfinal, mean = pimean, sd = sqrt(pivar), lower.tail = FALSE)
#> [1] 0.764337
```

The predictive probability for a positive final outcome is slightly below the originally planned power of 80%, but probably not sufficient amount to raise substantial concern.

11.6 Prediction intervals

Prediction intervals for a future test statistic or final treatment effect can be computed using the inverse of the predictive distribution for B_j given B_i, $1 \leq i < j \leq K$. We continue the above normal distribution example using Equation 11.5 and Equation 11.6. We will ignore intervening interim analysis decisions in this computation, which conditions on the results at the first interim and computes a 90% prediction interval. The inverse normal distribution `qnorm()` is applied as for the Bayes credible interval computation in Section 11.3.

```
bpi <- pimean + qnorm(c(0.05, 0.95)) * sqrt(pivar)
bpi
#> [1] 1.052884 4.422583
```

While the above computation is useful given a normal prior, we also have the `gsPI()` function to produce a prediction interval for any future analysis given an interim result. Note that the prediction for future analyses again ignores any intervening interim boundaries. The argument `i` is the interim we condition on, while `zi` is the test statistic at that analysis. The analysis we are predicting for is indicated by `j`, and the group sequential design by `x`. We first reproduce the interval above and then examine prediction intervals for the second and final analyses based on the first interim. Finally, the default `level = 0.95` produces a 95% prediction interval. By specifying `level = 0` we get a point estimate of the predicted outcome for the second interim analysis based on the first interim. While the second interim Z-value falls well within the prediction interval based on the first interim, it is quite different than the predicted outcome. Thus, the prediction interval suggests the differences between first and second analysis trends are consistent with variability early in the study.

```r
gsPI(
  level = 0.9,
  x = CAPTURE,
  i = 2,
  j = 3,
  zi = z[2],
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 1.052837 4.422774
```

```r
gsPI(
  level = 0.9,
  x = CAPTURE,
  i = 1,
  j = 2,
  zi = z[1],
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 1.925884 5.151808
```

```r
gsPI(
  level = 0.9,
  x = CAPTURE,
  i = 1,
  j = 3,
  zi = z[1],
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 2.182131 7.841837
```

```r
gsPI(
  level = 0,
  x = CAPTURE,
  i = 1,
  j = 2,
  zi = z[1],
  theta = prior$z,
  wgts = prior$wgts
)
#> [1] 3.538895
```

```
z[2]
#> [1] 1.925467
```

The prediction interval for the standardized treatment effect θ divides the above interval by $\sqrt{n_3}$. Since the analysis we are predicting for is the final analysis, $b_3 = t_3 z_3 = z_3$ and the prediction intervals are identical.

```
zpi <- bpi / sqrt(ti[3])
thetapi <- zpi / sqrt(CAPTURE$n.I[3])
zpi
#> [1] 1.052884 4.422583
```

```
thetapi
#> [1] 0.0276506 0.1161449
```

The prediction interval is quite wide which is perhaps counterintuitive given the the predictive probability 0.764 from the previous section.

11.7 Probability of success

The probability of a positive trial depends on the distribution of outcomes in the control and experimental groups. The probability of a positive trial given a particular parameter value θ was defined in

$$\alpha_i(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}, i = 1, 2, \ldots, k.$$

and

$$\alpha(\theta) \equiv \sum_{i=1}^{k} \alpha_i(\theta).$$

as

$$\alpha(\theta) = \sum_{i=1}^{K} P_\theta\{\{Z_i \geq b_i\} \cap_{j=1}^{i-1} \{a_j < Z_j < b_j\}\}.$$

If we consider θ to have a given prior distribution at the beginning of the trial, we can compute an unconditional probability of success for the trial. In essence, since we do not know if the experimental treatment works better than control treatment, we assign some prior beliefs about the likelihood that experimental is better than control and use those along with the size of the trial to compute the probability of success. The prior distribution for θ can be discrete or continuous. If the distribution is discrete, we define $m+1$ values $\theta_0 < \theta_2 \ldots < \theta_m$ and assign prior probabilities $P\{\theta = \theta_j\}$, $0 \leq j \leq m$ such that $\sum_{j=1}^{m} P\{\theta_j\} = 1$. The probability of success for the trial is then defined as

$$\text{POS} = \sum_{j=0}^{m} P\{\theta = \theta_j\}\alpha(\theta_j) \tag{11.7}$$

The simplest practical example is perhaps assuming a 2-point prior where the prior probability of the difference specified in the alternate hypothesis used to power the trial is p and the prior probability of no treatment difference is $1-p$. Suppose the trial is designed to have power $1-\beta = \alpha(\delta)$ when $\theta = \delta$ and Type I error $\alpha = \alpha(0)$ when $\theta = 0$. Then the probability of success for the trial is

$$\text{POS} = p \times (1-\beta) + (1-p) \times \alpha.$$

Assuming a 70% prior probability of a treatment effect δ, a 30% prior probability of no treatment effect, power of 90% and Type I error of 2.5% results in an unconditional probability of a positive trial of $0.7 \times 0.9 + 0.3 \times 0.025 = 0.6375$. In this case, it is arguable that POS should be defined as $0.7 \times 0.9 = 0.63$ since the probability of a positive trial when $\theta = 0$ should not be considered a success. This alternative definition becomes ambiguous when the prior distribution for θ becomes more diffuse. We will address this issue below in the discussion of the value of a trial design.

We consider a slightly more ambitious example and show how to use `gsProbability()` to compute Equation 11.7. We derive a design using `gsDesign()`, in this case using default parameters. We assume the possible parameter values are $\theta_i = i \times \delta$ where δ is the value of θ for which the trial is powered and $i = 0, 2, \ldots, 6$. The respective prior probabilities for these θ_i values are assumed to be 1/20, 2/20, 2/20, 3/20, 7/20, 3/20 and 2/20. We show the calculation and then explain in detail.

```r
y <- gsDesign()
theta <- y$theta[2] * array(0:6) / 4
ptheta <- c(1, 2, 2, 3, 7, 3, 2) / 20
x <- gsProbability(theta = theta, d = y)
one <- array(1, 3)
pos <- one %*% x$upper$prob %*% ptheta
pos
```

```
#>             [,1]
#> [1,] 0.7136783
```

Note that `theta[2]` is the value δ for which the trial is powered as noted in the first example in the introduction Section 4.1. The last 4 lines can actually be replaced by the function `POS`: `gsPOS(x, theta, ptheta)`. For those not familiar with it `%%` represents matrix multiplication, and thus the code `one %% x$upper$prob %% ptheta` is doing the computation

$$\sum_{j=0}^{m} P\{\theta_j\} \sum_{i=0}^{K} \alpha_i(\theta_j).$$

For a prior distribution that is continuous with density $f(\theta)$ we define

$$\text{POS} = \int_{-\infty}^{\infty} f(\theta)\alpha(\theta)d\theta. \tag{11.8}$$

Numerical integration is required to implement this calculation, but we can still use the `pos()` function just defined. For instance, assuming $\theta \sim N(\mu = \delta, \sigma^2 = (\delta/2)^2)$ we can use `normalGrid()` from the gsDesign package to generate a grid and normal densities on that grid that can be used to perform numerical integration. For this particular case

```
y <- gsDesign()
delta <- y$theta[2]
g <- normalGrid(mu = delta, sigma = delta / 2)
plot(g$z, g$wgts, main = "Integration weights for normal density.")
```

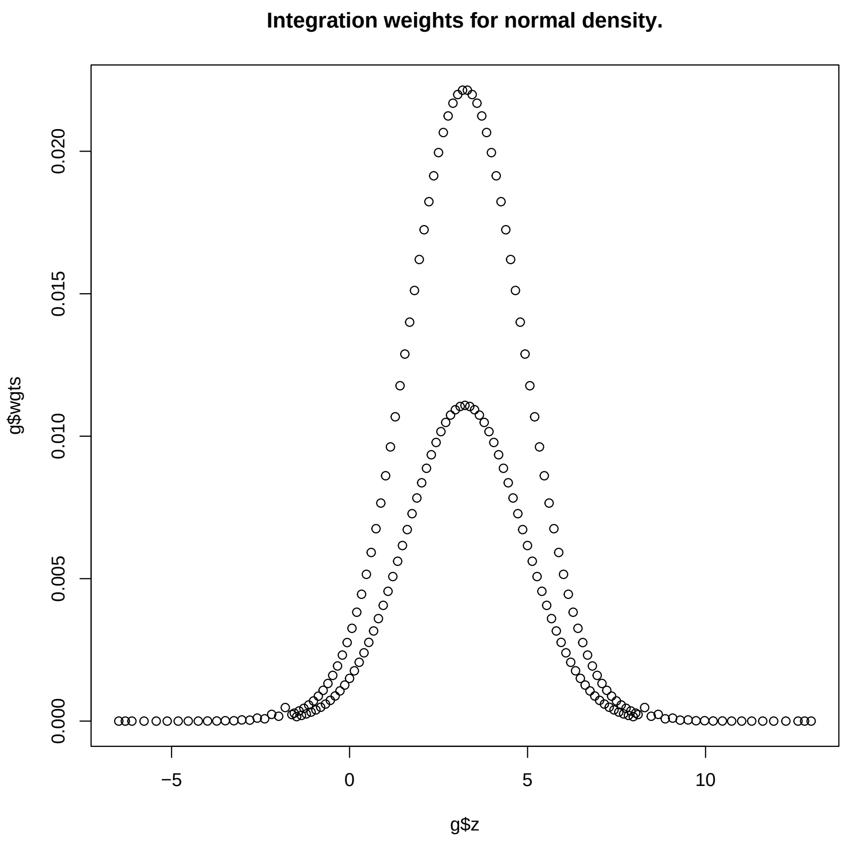


Figure 11.6

```
gsPOS(y, g$z, g$wgts)
#> [1] 0.7484896
```

This computation yields a probability of success of 0.748. The `normalGrid()` function is a lower-level function used by `gsProbability()` and `gsDesign()` that is normally obscured from the user. For Bayesian computations with a normal prior distribution, such as here, it can be quite useful as in the above example. The values returned above in `g$wgts` are the normal density multiplied by weights generated using Simpson's rule. The plot generated by the above code

```
g <- normalGrid(mu = 0, sigma = 1)
plot(g$z, g$wgts)
```

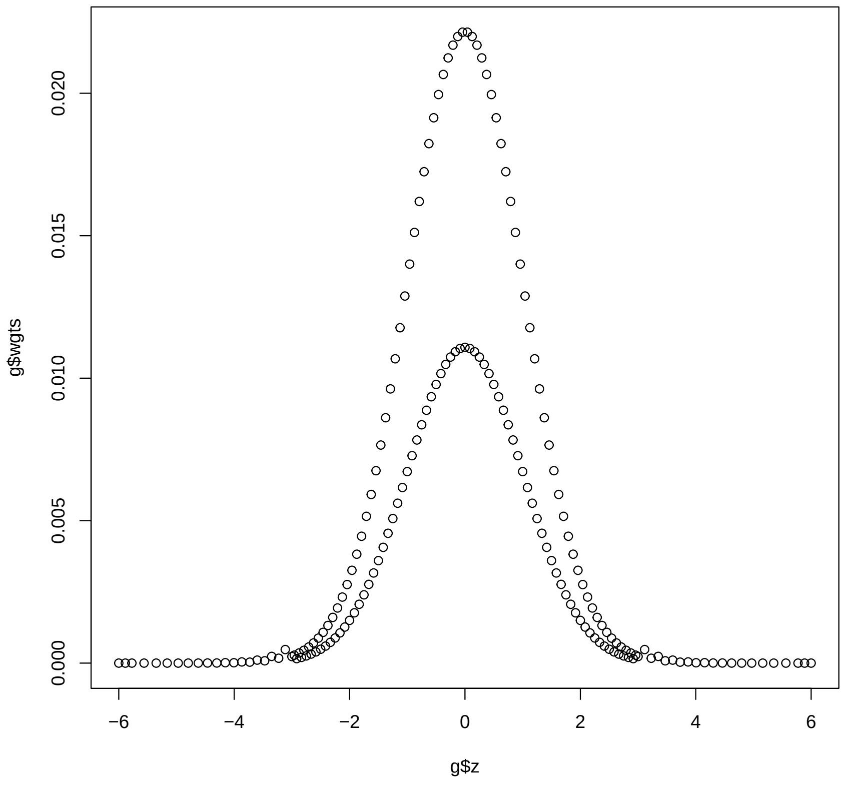


Figure 11.7

shows that these values alternate as higher and lower values about a smooth function. If you compute `sum(g$wgts)` you will confirm that the approximated integral over the real line of this density is 1, as desired.

To practice with this, assume a more pessimistic prior with $\mu = \sigma = \delta/2$ to obtain a probability of success of 0.428.

We generalize Equation 11.7 and Equation 11.8 by letting $F()$ denote the cumulative distribution function for θ and write

$$\text{POS} = \int_{-\infty}^{\infty} \alpha(\theta) dF(\theta).$$

This notation will be used in further discussions to provide formulas applicable to both continuous and discrete distributions.

11.8 Updating probability of success based on blinded results

Futility bounds are often set up to be informative about emerging treatment effects. That is, a positive trend is often required to pass a futility bound. Efficacy bounds usually are only informative to a lesser extent, but knowing that an efficacy bound has not been crossed at an interim analysis generally rules out an extremely positive effect after early interim analyses and a moderately positive effect later in the trial. Thus, knowing that a trial has not crossed a futility or efficacy bound at an interim analysis can be helpful in updating the probability of success we have computed above. In this subsection we will restrict consideration to the probability of ultimate trial success. For $1 \leq i < K$ we denote the event that no boundary has been crossed at or before analysis i by

$$A_i = \cap_{j=1}^{i-1}\{a_j < Z_j < b_j\}$$

The probability of observing A_i is

$$P\{A_i\} = 1 - \int \sum_{j=1}^{i} (\alpha_j(\theta) + \beta_j(\theta)) dF(\theta)$$

Letting B denote the event that the trial crosses an upper bound at or before the end of the trial and before crossing a lower bound compute

$$P\{A_i \cap B\} = \int \sum_{j=i+1}^{K} \alpha_j(\theta) dF(\theta)$$

Based on these 2 equations, we can now compute for $1 \leq i < K$ the conditional probability of a positive trial given that no boundary has been crossed by interim i as

$$P\{B|A_i\} = P\{A_i \cap B\}/P\{A_i\}.$$

Calculations for the 2 probabilities needed are quite similar to the `gsPOS()` function considered in the previous subsection. The conditional probability of success is computed using the function `gsCPOS()`. For the case considered previously with $\theta \sim N(\mu = \delta, \sigma = \delta/2)$ and a default design we had a probability of success of 0.748. The following code shows that if neither trial boundary is crossed at the first interim, the updated (posterior) probability of success is 0.733. After 2 analyses, the posterior probability of success is 0.669.

```r
y <- gsDesign()
delta <- y$theta[2]
g <- normalGrid(
  bounds = c(-30, 30) * delta / 2,
  mu = delta,
  sigma = delta / 2
)
gsPOS(x = y, theta = g$z, wgts = g$wgts)
#> [1] 0.7484896
```

```r
gsCPOS(1, y, theta = g$z, wgts = g$wgts)
#> [1] 0.7331074
```

```r
gsCPOS(2, y, theta = g$z, wgts = g$wgts)
#> [1] 0.6688041
```

To ensure a higher conditional probability of success for the trial, a more aggressive futility bound could be employed at the expense of requiring an increased sample size to maintain the desired power. The code `y$n.I` shows that the default design requires an inflation factor of 1.07 for the sample size compared to a fixed design with the same Type I error and power. By employing an aggressive Hwang-Shih-DeCani spending function with $\gamma = 1$ for the futility bound, the sample size inflation factor is increased to 1.23 ((y <- gsDesign(sflpar=1))). The probability of success for this design at the beginning of the trial using the same prior distribution as above is still 0.748, but the conditional probability of success after not hitting a boundary by interim 1 (interim 2) is now 0.788 (0.761). While there are many other considerations in choosing a futility bound and other prior distributions give other results, this example suggests that something more aggressive than the default futility bound in `gsDesign()` may be desirable.

11.9 Calculating the value of a clinical trial design

Here we generalize the concept of the probability of success of a trial given above to the value of a trial. We assume that a trial that stops with a positive result with information I_i at analysis i of a trial when the true treatment effect is θ can be given by a function $u(\theta, I_i)$, $1 \leq i \leq K$. Now the formula for probability of success can be generalized to

$$U = \int_{-\infty}^{\infty} f(\theta) \sum_{i=1}^{K} \alpha_i(\theta) u(\theta, I_i) d\theta. \tag{11.9}$$

Note that the value above is only assumed to depend on whether or not a boundary is crossed and when it is crossed, not the actual treatment effect at the time the study stops and not other factors, such as safety, or continuing the trial when a boundary has been crossed. A more general formula that incorporates a costs that are incurred whether or not a trial is positive. If this formula also discounted future costs and benefits to present-day values, it would be termed a net present value and can be defined in a simplified form as shown below. Here we assume that the final upper and lower bounds are the same so that a bound is crossed with probability 1 at some point in the trial. This formulation

$$\text{NPV} = \int_{-\infty}^{\infty} f(\theta) \sum_{i=1}^{K} \left[\alpha_i(\theta) u(\theta, I_i) - (\alpha_i(\theta) + \beta_i(\theta)) c(\theta, I_i)\right] d\theta.$$

The function below computes the integral Equation 11.9. For this implementation, must be a scalar, a vector of length or a matrix of the same dimension as (rows and columns) rather than a function.

```r
value <- function(x, theta, wgts, u, c) {
  x <- gsProbability(theta = theta, d = x)
  one <- array(1, x$k)
  totprob <- x$upper$prob + x$lower$prob
  for (i in 1:length(x$theta)) {
    totprob[x$k, i] <- totprob[x$k, i] + (1 - sum(totprob[, i]))
  }
  as.numeric(one %*% (u * x$upper$prob - c * totprob) %*% wgts)
}
```

We now consider the CAPTURE trial with the planned interim analyses previously planned and the previously specified prior distribution for θ. We further assume a “utility” of a positive trial is linear in sample size and equal to 900-n/3. The corresponding costs, including opportunity costs, are assumed to be 30+n/15. These cost assumptions are simple and arbitrary, primarily meant to inform as an example. The net value is found to be 234.1.

```r
value(
  x = CAPTURE,
  theta = prior$z,
  wgts = prior$wgts,
  u = 900 - CAPTURE$n.I / 3,
  c = 30 + CAPTURE$n.I / 15
)
#> [1] 234.0738
```

11.10 Optimization example: selecting a futility bound

We finish with an example computing a futility bound that optimizes the value of a design. We will assume the spending function for the efficacy bound is fixed and and will select an optimal γ for a Hwang-Shih-DeCani spending function for the lower bound. We allow the user to specify the number of interim analyses as well as the desired Type I and Type II error and the prior distribution for the treatment effect. The function provides the value of a trial that stops for a positive result after enrolling patients when the true treatment effect is . The previous value function is coded as:

```
valfn <- function(n.I, theta, vcon, ccon) {
  return(list(
    u = vcon[1] + n.I * vcon[2],
    c = ccon[1] + n.I * ccon[2]
  ))
}
vcon <- c(900, -1 / 3)
ccon <- c(30, 1 / 15)
```

We now write a function to compute the value of designs where the only thing changing between design inputs is the single-parameter for the lower-bound spending function. Since the optimization function we will apply will minimize the objective function, we let change the sign of the value when we output the computed value of a design below.

```
lbValue <- function(
    x = -2,
    k = 3,
    test.type = 4,
    alpha = 0.025,
    beta = 0.1,
    astar = 0,
    delta = 0,
    n.fix = 1,
    timing = 1,
    sfu = sfHSD,
    sfupar = -3,
    sfl = sfHSD,
    tol = 1e-06,
    r = 18,
    f, theta, wgts, vcon, ccon) {
  d <- gsDesign(
    sflpar = x,
    k = k,
    test.type = test.type,
    alpha = alpha,
    beta = beta,
    delta = delta,
    n.fix = n.fix,
    timing = timing,
    sfu = sfu,
    sfupar = sfupar,
    sfl = sfl,
    tol = tol,
    r = r
  )
  val <- f(d$n.I, theta, vcon = vcon, ccon = ccon)
  -value(x = d, theta = theta, wgts = wgts, u = val$u, c = val$c)
}
```

We first compute this value function for `sfupar = -2`, as before. Note since we have not restricted the interims to be a 350 and 700 patients, the value is slightly different than before with the sample sizes shown below.

```
n.fix <- 1
lbValue(
  x = -2,
  k = 3,
  beta = 0.2,
  n.fix = n.fix,
  timing = c(0.25, 0.5),
  sfupar = -3,
  f = valfn,
  theta = prior$z,
  wgts = prior$wgts,
  vcon = vcon,
  ccon = ccon
)
#> [1] 5.133372
```

```
ceiling(
  gsDesign(
    k = 3,
    timing = c(0.25, 0.5),
    sfupar = -3,
    beta = 0.2,
    n.fix = n.fix
  )$n.I / 2
) * 2
#> [1] 2 2 2
```

We can now find an optimal lower Hwang-Shih-DeCani bound and sample size for the CAPTURE trial with interims after 25% and 50% of patients enrolled. This is done by calling the R optimization function `nlminb()` inputting the appropriate parameters from above. We save the output from `nlminb()` in the variable `best`. The expected value of the optimal design is not terribly different from what we computed previously. This is in spite of a fairly different and more conservative lower bound.

```
best <- nlminb(
  start = -2,
  objective = lbValue,
  k = 3,
  beta = 0.2,
  n.fix = n.fix,
  timing = c(0.25, 0.5),
  sfupar = -3,
  f = valfn,
  theta = prior$z,
  wgts = prior$wgts,
  vcon = vcon,
  ccon = ccon
)
-best$objective
#> [1] -4.133656
```

```
best$par
#> [1] -11.21022
```

```
OPTCAPTURE <- gsDesign(
  sflpar = best$par,
  k = 3,
  beta = 0.2,
  n.fix = n.fix,
  timing = c(0.25, 0.5),
  sfupar = -3
)
OPTCAPTURE$n.I
#> [1] 0.2551781 0.5103561 1.0207123
```

```
OPTCAPTURE$lower$bound
#> [1] -2.517904 -1.190569  2.001880
```

The general recommendation is to consider optimal designs to ensure that the final design you choose based on a variety of criteria provides a “value” that is reasonably close to optimal. In the case of CAPTURE, the previously planned bound that was more aggressive might be considered reasonably close to the optimal design and therefore acceptable.

References

Keaven M. Anderson. Optimal spending functions for asymmetric group sequential designs. *Biometrical Journal*, 49(3):337–345, 2007.

Keaven M. Anderson and Jason B. Clark. Fitting spending functions. *Statistics in Medicine*, 29(3):321–327, 2010.

Peter Bauer and F. Kohne. Evaluation of experiments with adaptive interim analyses. *Biometrics*, 50(4):1029–1041, 1994.

Center for Drug Evaluation and Research. Guidance for industry on diabetes mellitus-evaluating cardiovascular risk in new antidiabetic therapies to treat type 2 diabetes. Technical report, United States Department of Health and Human Services, Food and Drug Administration, 2008. URL https://www.federalregister.gov/d/E8-30086.

Lu Cui, HM James Hung, and Sue-Jane Wang. Modifications of sample size in group sequential clinical trials. *Biometrics*, 55(3):853–857, 1999.

Inc. Cytel. *EAST 5*. Cytel, Inc., Cambridge, MA, 2007.

Conor P Farrington and Godfrey Manning. Test statistics and sample size formulae for comparative binomial trials with null hypothesis of non-zero risk difference or non-unity relative risk. *Statistics in Medicine*, 9(12): 1447–1454, 1990.

Ian Gordon and Ray Watson. The myth of continuity-corrected sample size formulae. *Biometrics*, 52(1):71–76, 1985.

Irving K Hwang, Weichung J Shih, and John S De Cani. Group sequential designs using a family of type 1 error probability spending functions. *Statistics in Medicine*, 9(12):1439–1445, 1990.

Christopher Jennison and Bruce W. Turnbull. *Group Sequential Methods with Applications to Clinical Trials*. Chapman and Hall/CRC, Boca Raton, FL, 2000.

Kyungmann Kim and David L Demets. Design and analysis of group sequential tests based on type I error spending rate functions. *Biometrika*, 74(1): 149–154, 1987.

John M. Lachin and Mary A. Foulkes. Evaluation of sample size and power for analyses of survival with allowance for nonuniform patient entry, losses to follow-up, noncompliance, and stratification. *Biometrics*, 42(3):507–519, 1986.

K K Gordon Lan and David L DeMets. Discrete sequential boundaries for clinical trials. *Biometrika*, 70(3):659–663, 1983.

Qing Liu and George YH Chi. On sample size and inference for two-stage adaptive designs. *Biometrics*, 57(1):172–177, 2001.

Richard W Madsen and Kenneth B Fairbanks. p values for multistage and sequential tests. *Technometrics*, 25(3):285–293, 1983.

Ollie Miettinen and Markku Nurminen. Comparative analysis of two rates. *Statistics in Medicine*, 4(2):213–226, 1985.

Hans-Helge Müller and Helmut Schäfer. Adaptive group sequential designs for clinical trials: combining the advantages of adaptive and of classical group sequential approaches. *Biometrics*, 57(3):886–891, 2001.

Peter C O'Brien and Thomas R Fleming. A multiple testing procedure for clinical trials. *Biometrika*, 35(3):549–556, 1979.

Sandro Pampallona and Anastasios A Tsiatis. Group sequential trials for one-sided and two-sided hypothesis testing with provision for early stopping in favor of the null hypothesis. *Journal of Statistical Planning and Inference*, 42(1-2):19–35, 1994.

Stuart J. Pocock. Group sequential methods in the design and analysis of clinical trials. *Biometrika*, 64(2):191–199, 1977.

Michael A Proschan and Sally A Hunsberger. Designed extension of studies based on conditional power. *Biometrics*, 51(4):1315–1324, 1995.

Michael A. Proschan, K K Gordon Lan, and Janet Turk Wittes. *Statistical Monitoring of Clinical Trials. A Unified Approach.* Springer, New York, NY, 2006.

Daniel O Scharfstein, Anastasios A Tsiatis, and James M Robins. Semiparametric efficiency and its implication on the design and analysis of group-sequential studies. *Journal of the American Statistical Association*, 92(440): 1342–1350, 1997.

Thomas Sellke and David Siegmund. Sequential analysis of the proportional hazards model. *Biometrika*, 70(2):315–326, 1983.

Eric Slud and LJ Wei. Two-sample repeated significance tests based on the modified Wilcoxon statistic. *Journal of the American Statistical Association*, 77(380):862–868, 1982.

The CAPTURE Investigators. Randomised placebo-controlled trial of abciximab before and during coronary intervention in refractory unstable angina: the CAPTURE study. *The Lancet*, 349(9063):1429–1435, 1997.

The GUSTO V Investigators. Reperfusion therapy for acute myocardial infarction with fibrinolytic therapy or combination reduced fibrinolytic therapy and platelet glycoprotein IIb/IIIa inhibition: the GUSTO V randomised trial. *The Lancet*, 357(9272):1905–1914, 2001.

Anastasios A Tsiatis. Repeated significance testing for a general class of statistics use in censored survival analysis. *Journal of the American Statistical Association*, 77(380):855–861, 1982.

Samuel K Wang and Anastasios A Tsiatis. Approximately optimal one-parameter boundaries for group sequential trials. *Biometrics*, 43(1):193–199, 1987.