

gsDesign: An R Package for Designing Group Sequential Clinical Trials Version 1.10 Manual

Keaven M. Anderson, Dan (Jennifer) Sun, Zhongxin (John) Zhang
Merck Research Laboratories

September 9, 2007

Abstract

The gsDesign package provides support for standard and customized group sequential clinical trial designs. 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. 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. This enables the user to easily create designs with features they desire, such as designs with minimum expected sample size.

Contents

| | | |
|----------|--|----------|
| 1 | Overview | 2 |
| 2 | Basic Features | 2 |
| 3 | Installation and Online Help | 3 |
| 4 | Syntax | 4 |
| 4.1 | gsDesign() syntax | 4 |
| 4.2 | gsProbability() syntax | 6 |
| 5 | The gsDesign and gsProbability Object Types | 7 |
| 5.1 | The spendfn Class | 7 |
| 5.2 | The gsProbability Class | 8 |
| 5.3 | The gsDesign Class | 8 |
| 6 | Formatted Output | 9 |
| 7 | Spending Functions | 9 |
| 7.1 | Spending Function Basics | 10 |
| 7.2 | Spending Function Details | 11 |
| 7.3 | Investigational Spending Functions | 13 |
| 7.4 | Optimized Spending Functions | 14 |
| 7.5 | Writing Code for a New Spending Function | 16 |

| | |
|------------------------------|-----------|
| 8 Detailed Examples | 18 |
| 9 Statistical Methods | 29 |
| 10 Contacts | 33 |

1 Overview

Two R functions are supplied to provide basic computations related to designing group sequential clinical trials.

1. The first function, `gsDesign()`, 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. While `gsDesign()` develops a design and summarizes its properties under the null and alternative hypothesis, the second function, `gsProbability()`, computes boundary crossing probabilities and expected sample size of a design for arbitrary user-specified treatment effects.

The package design strategy should make these routines 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()`. This should make it easy to incorporate design specification and properties into documents, as required.

Functions are set up to be called directly from the R command line. Default arguments and output for `gsDesign()` are included to make initial use simple. Sufficient options are available, however, to make the routine very flexible. The reader may wish to read BASIC FEATURES and then jump immediately to DETAILED EXAMPLES to get a feel for how the routines work. To try the routines out, read also INSTALLATION AND ONLINE HELP. These three sections along with online help may enable the reader to develop a design quickly without reading the full specification in SYNTAX, THE `gsDesign` AND `gsProbability` OBJECT CLASSES, FORMATTED OUTPUT, and SPENDING FUNCTIONS. For those interested in the theory behind this package, the section titled STATISTICAL METHODS provides background.

2 Basic Features

There are several key design features common to both `gsDesign()` and `gsProbability()`:

1. Computations are based on asymptotic approximations as provided by Jennison and Turnbull [3].
2. Power plots and boundary plots are available, in addition to various printing formats for summarization.
3. Rather than supporting a wide variety of endpoint or design types (e.g., normal, binomial, time to event), the `gsDesign()` routine allows input of the sample size for a fixed design with no interim analysis and adjusts the sample size appropriately for a group sequential design.

In addition, the following apply to `gsDesign()`:

1. Two-sided symmetric and asymmetric designs are supported, as well as one-sided designs.

2. The spending function approach to group sequential design first published by Lan and DeMets [5] is implemented. Commonly used spending functions published by Hwang, Shih, and DeCani [2] and by Kim and DeMets [4] are provided. Other built-in spending functions are included. Two- and three parameter spending functions are particularly flexible. There is also point-wise specification of spending available. Finally, specifications are given for users to write their own spending functions.
3. As an alternative to the spending function approach, the Wang and Tsatis [8] family of boundaries is also available for symmetric or one-sided designs. This family includes O'Brien-Fleming and Pocock boundaries as members.
4. For asymmetric designs, lower bound spending functions may be used to specify lower boundary crossing probabilities under the alternative hypothesis (beta spending) or null hypothesis (recommended when number of analyses is large or when faster computing is required – e.g., for optimization).
5. Normally it is assumed that when a boundary is crossed at the time of an analysis, the clinical trial must stop without a positive finding. In this case, the boundary is referred to as binding. For asymmetric designs, a user option is available to ignore lower bounds when computing Type I error. Under this assumption the lower bound is referred to as non-binding. That is, the trial may continue rather than absolutely requiring stopping when the lower bound is crossed. This is a conservative design option sometimes requested by regulators to preserve Type I error when they assume a sponsor may choose to ignore an aggressive futility (lower) bound if it is crossed.

3 Installation and Online Help

The package comes in a binary format for a Windows platform in the file `gsDesign-1.10.zip` (may be updated to fix bugs in a file such as `gsDesign-1.10-1.zip`). This file includes a copy of this manual in the file `gsDesignManual.pdf`. The source, available for other platforms (but only tested minimally!) is in the file `gsDesign-1.10.tar.gz` (may be updated to fix bugs in a file such as `gsDesign-1.10-1.tar.gz`). Following are basic instructions for installing the binary version on a Windows machine. It is assumed that a 'recent' version of R is installed. From the Windows interface of R, select the Packages menu line and from this menu select Install packages from local zip files... Browse to select `gsDesign-1.10.zip`. Once installed, you need to load the package by selecting the Packages menu line, selecting Load package... from this menu, and then selecting `gsDesign`. You are now ready to use the routines in the package.

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

```
> 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. In the Window version of R, this will bring up a "Contents" window and a documentation window. In the contents window, open the branch of documentation headed by "Package `gsDesign`: Titles." The help files are organized sequentially under this heading. As noted above, those interested in a quick start may choose to read DETAILED EXAMPLES and try out the package before coming back to read more detailed specifications below.

4 Syntax

The two primary functions available in this package are:

- `gsDesign(argument1= value1, ...)`
- `gsProbability(argument1= value1, ...)`

All arguments are in lower case with the exception of `n.I` in `gsProbability()`. Although there are many arguments available in `gsDesign()`, many of these need not be specified as the defaults will be adequate. Returned values from `gsDesign()` and `gsProbability()` are objects from newly defined classes named `gsDesign` and `gsProbability`. The `gsDesign` class is an extension of (inherits the characteristics of) the `gsProbability` class. These classes are described in the next section, THE `gsDesign` AND `gsProbability` OBJECT CLASSES. The output functions `print()`, and `plot()` for the `gsDesign` and `gsProbability` classes are described further in FORMATTED OUTPUT.

In the following, there is a single parameter θ for which we generally are trying to test a null hypothesis $H_0: \theta = 0$ against some alternative such as $H_1: \theta \neq 0$. In general, the parameter θ will be a standardized treatment difference and the statistic Z_n for testing after n observations have a distribution that is well approximated by a normal distribution with mean $\sqrt{n\theta}$ and variance 1. See Jennison and Turnbull [3] for extensive discussions of various types of endpoints for which the approximation is reasonable. For example, normal, binomial and time-to-event endpoints may be considered. STATISTICAL METHODS provides more detail.

4.1 `gsDesign()` syntax

The most general form of the call to `gsDesign()` with default arguments is:

```
gsDesign(k=3, test.type=4, alpha=0.025, beta=0.1, astar=0, delta=0, n.fix=1,
        timing=1, sfu=sfHSD, sfupar= -4, sfl=sfHSD, sflpar= -2, tol=0.000001, r=18).
```

The arguments are as follows:

- **k** = integer ($k > 1$). Specifies the number of analyses, including interim and final. The default value is 3. NOTE: Use of very large **k** will produce an error message. How large varies with the value of **test.type** specified. For example, with default arguments (**test.type**=3), specifying **k** > 23 will produce the error message "False negative rates not achieved." For other values of **test.type**, the upper limit is larger.
- **test.type** = integer ($1 \leq \text{test.type} \leq 6$). In the following we will denote by θ the parameter we are testing for. The null hypothesis will always be $H_0: \theta = 0$. The alternative hypothesis of interest varies with the value of **test.type**.
 - =1:** One-sided testing. Tests the null hypothesis $H_0: \theta = 0$ against the alternative hypothesis $H_1: \theta > 0$. User specifies only upper boundary crossing probabilities under the null hypothesis. There is no lower boundary.
 - =2:** Symmetric, 2-sided testing. Tests $H_0: \theta = 0$ against the alternative hypothesis $H_1: \theta \neq 0$. Boundary crossing probabilities are specified under the null hypothesis. User specifies only upper boundary crossing probabilities
 - =3:** Asymmetric 2-sided testing with binding lower bound and beta spending. Asymmetric 2-sided testing is also often referred to as two one-sided tests in that testing is done both for efficacy (attempts to reject $H_0: \theta = 0$ in favor of $H_1: \theta > 0$) and futility (attempts to reject $H_1: \theta = \text{delta}$ in favor of $H_0: \theta < \text{delta}$). Upper and lower boundary crossing

probabilities are asymmetric. Upper boundary crossing probabilities are specified under $H_0: \theta = 0$ (α -spending). Lower boundary crossing probabilities are specified under $H_1: \theta = \text{delta}$ (β -spending). Computation of upper boundary crossing probabilities under the null hypothesis (Type I error) assumes the lower boundary is binding; that is, the trial MUST stop if the lower boundary is crossed. See STATISTICAL METHODS for details on boundary crossing probabilities for options 3 through 6.

- =4:** Default. Asymmetric 2-sided testing with non-binding lower bound and beta spending. Same as `test.type=3`, except that Type I error computation assumes lower boundary is non-binding; i.e., the calculation assumes that the trial will only stop if upper boundary is crossed – it will continue if the lower boundary is crossed. Type II error computation assumes the trial will stop when either boundary is crossed. The effect of this is to raise the upper boundaries after the 1st interim analysis relative to `test.type=3`. See STATISTICAL METHODS for details.
 - =5:** Asymmetric 2-sided testing with binding lower bound and lower bound spending specified under the null hypothesis. Same as `test.type=3`, except that lower boundary crossing probabilities are specified under $H_0: \theta = 0$. See STATISTICAL METHODS for details.
 - =6:** Asymmetric 2-sided testing with non-binding lower bound and lower bound spending specified under the null hypothesis. Same as `test.type=4`, except that lower boundary crossing probabilities are specified under $H_0: \theta = 0$. See STATISTICAL METHODS for details.
- **alpha** = probability value (real; $0 < \text{alpha} < 1$; for `test.type=2`, must have $\text{alpha} < 0.5$). Specifies the total Type I error summed across all analyses. For all design types (including symmetric, 2-sided) this is the probability of crossing the upper boundary under the null hypothesis. The default value is 0.025. For symmetric 2-sided testing (`test.type=2`), this translates into 0.05 for the combined total probability of crossing a lower or upper bound at any analysis. See STATISTICAL METHODS for details.
 - **beta** = probability value (real; $0 < \text{beta} < 1 - \text{alpha}$). Specifies the total Type II error summed across all analyses. The default value is 0.1, which corresponds to 90% power.
 - **astar** = probability value (real; $0 \leq \text{astar} \leq 1 - \text{alpha}$). Normally not specified. If `test.type=5` or 6, **astar** specifies the probability of crossing a lower bound at all analyses combined. This will be changed to $1 - \text{alpha}$ when default value of 0 is used. Since this is the expected usage, **astar** will normally not be specified by the user.
 - **delta** = real value ($\delta > 0$). This is the standardized effect size used for alternative hypothesis for which the design is powered. If **delta** > 0 , sample size is determined using this effect size (see STATISTICAL METHODS for details); If the default **delta** = 0 is given, sample size is determined by **n.fix** as noted below. Only one of **delta** and **n.fix** need be specified by the user.
 - **n.fix** = real value (**n.fix** > 0). If **delta** > 0 then **n.fix** is ignored. For the default values (**delta**=0 and **n.fix**=1) the returned values (R_1, \dots, R_K) for sample sizes are actually sample size ratios compared to a fixed design with no interim analysis, where the denominator for all of these ratios is the sample size for a fixed design with no interim analysis based on input values of **alpha** and **beta**. If **delta**=0 and **n.fix** > 1 , think of **n.fix** as the fixed sample size of a study without interim analysis for the given values of **alpha** and **beta**. `gsDesign()` will then inflate this value to get a maximum sample size to give the specified Type II error/power for the specified group sequential design. We will also denote **n.fix** as N_{fix} below. See STATISTICAL METHODS for details on how to calculate sample size.

- **timing** = 1 or $c(t_1, \dots, t_{k-1})$ or $c(t_1, \dots, t_k)$ where $0 < t_1 < \dots < t_k = 1$. Specifies the cumulative proportionate timing for analyses; e.g., $t_2=0.4$ means the second interim will include the first 40% of planned observations. For equal spacing, **timing**=1 (default), the timing will be determined by number of interim looks **k**.
- **sfu** = spending function (default = **sfHSD**). The parameter **sfu** specifies the upper boundary using a spending function. For one-sided and symmetric two-sided testing (**test.type**=1, 2), **sfu** is the only spending function required to specify spending. The default value is **sfHSD** which is a Hwang-Shih-DeCani spending function. See SPENDING FUNCTIONS for options available.
- **sfupar** = real value (default = -4) The parameter **sfupar** specifies any parameters needed for the spending function specified by **sfu**. The default of -4 when used with the default **sfu**=**sfHSD** provides a conservative, O'Brien-Fleming-like upper bound. For spending functions that do not require parameters this will be ignored. Again, see SPENDING FUNCTIONS.
- **sfl** = spending function (default = **sfHSD**). Specifies the spending function for lower boundary crossing probabilities. The parameter **sfl** is ignored for one-sided testing (**test.type**=1) or symmetric 2-sided testing (**test.type**=2).
- **sflpar** = real value (default = -2). The parameter **sflpar** specifies any parameters needed for the spending function specified by **sfl**. The default value of -2 when used with the default **sfl**=**sfHSD** provides a somewhat conservative lower bound. For spending functions that do not require parameters this will be ignored.
- **tol** = value. Specifies the stopping increment for iterative root-finding algorithms used to derive the user-specified design. The default value is 0.000001; must be >0 and ≤ 0.1 . This should probably be ignored by the user, but is provided as a tuning parameter.
- **r**=positive integer up to 80 (default = 18). This is a parameter used to set the grid size for numerical integration computations; see Jennison and Turnbull [3], Chapter 19. This should probably be ignored by the user, but is provided as a tuning parameter.

4.2 gsProbability() syntax

The generic call to **gsProbability()** is of the form:

```
gsProbability(k = 0, theta, n.I, a, b, r = 18, d=NULL)
```

The value of **theta** must always be specified. This function is designed to be called in one of two ways. First, using a returned value from a call to **gsDesign()** in the input parameter **d**. If **d** is non-null, the parameterization specified there determines the output rather than the values of **k**, **n.I**, **a**, **b**, and **r**. If **k** is not the default of 0, the user must specify the set of input variables **k**, **n.I**, **a**, **b**, and **r**. In the latter form, the arguments **n.I**, **a**, and **b**, must all be vectors with a common length **k** equaling the number of analyses, interim and final, in the design.

The arguments are as follows:

- **k** – non-negative integer (default = 0). The number of analyses, including interim and final; default value of 0. MUST be 0 if the argument **d** is non-null.
- **theta**=vector of real values. This specifies parameter values (standardized effect sizes) for which boundary crossing probabilities and expected sample sizes are desired.
- **n.I** - vector of length **k** of real values $0 < I_1 < I_2 \dots < I_k$. Specifies the statistical information I_1, \dots, I_k (see STATISTICAL METHODS) for each analysis.

- **a** - vector of length **k** of real values (l_1, \dots, l_k) . Specifies lower boundary values.
- **b** - vector of length **k** of real values (u_1, \dots, u_k) . Specifies upper boundary values. For $i < k$ should have $l_i < u_i$. For $i = k$, should have $l_i \leq u_i$.
- **r** - positive integer up to 80 (default = 18). Same as for `gsDesign()`. This should probably be ignored by the user, but is provided as a tuning parameter.
- **d** - a returned value from a call to `gsDesign()`; values of **k**, **n.I**, **a**, **b**, and **r** will be taken from **d** when the input value of **k** is 0. Note that if **d** is specified and **k**=0, then the returned value will be an object of type `gsDesign`; otherwise, the type returned will be the simpler `gsProbability` class.

5 The `gsDesign` and `gsProbability` Object Types

This package creates three object (storage) classes to store the structured information returned by `gsDesign()`, `gsProbability()` and spending function routines. Thus, a single variable name can be used to access, print or plot a set of returned values.

5.1 The `spendfn` Class

The first class, `spendfn`, is used to store information about either the upper or the lower bound of a design: the spending function, z-values for the stopping bounds, spending at each analysis and boundary crossing probabilities under parameter values of interest. It contains the following elements:

- **sf** - a spending function or character value. Normally this will be a spending function, but if Wang-Tsiatis ("WT"), Pocock ("Pocock"), or O'Brien-Fleming ("OF") are specified (these are not spending function designs), then **sf** will be a character string. This is useful if you later wish to plot the spending function for a larger set of values than specified in an original call. (NOTE for writers of new spending functions: this is not set in the spending function itself, but in `gsDesign()`)
- **name** - a character string specifying the spending function used.
- **param** - parameter value or values to fully specify the spending function using the spending function family specified in **sf**.
- **parname** - a text string or vector of text strings (matching the length of **par**) supplying the names of parameters.
- **spend** - a vector containing the amount of alpha- or beta-spending at each analysis; this is determined in a call to `gsDesign()`.
- **bound** - this is null when returned from the spending function, but is set in `gsDesign()` if the spending function is called from there. Contains z-values for bounds of a design.
- **prob** - this is null when returned from the spending function, but is set in `gsDesign()` if the spending function is called from there. Contains a $k \times n$ matrix of probabilities of boundary crossing at *i*-th analysis for *j*-th **theta** value in **prob[i,j]**. Upon return from `gsDesign()`, **n**=2 and the values of **theta** considered are 0 and **delta**. More values of **theta** can be added by a call to `gsProbability()`. Columns correspond to the values specified by **theta**. All boundary crossing is assumed to be binding for this computation; that is, the trial must stop if a boundary is crossed.

- **errcode** – 0 if no error in spending function specification, > 0 otherwise.
- **errmsg** – a text string corresponding to **errcode**; for no error, this is "No error."

5.2 The gsProbability Class

The second object class is gsProbability. The members of this class on output from **gsProbability()** are: **k**, **n.I**, **lower**, **upper**, **r**, **theta**, **errcode**, and **errmsg**. Most of these are described in the input to **gsProbability()**. The exceptions are as follows:

- **upper** – on output, contains information on upper spending; this is a member of the spendfn class described above. Values of **upper** returning in a gsProbability class will only contain the elements **bound** and **prob**.
- **lower** – on output, contains information on lower spending; this is a member of the spendfn class described above. Values of **lower** returning in a gsProbability class will only contain the elements **bound** and **prob**.
- **en** – a vector of the same length as **theta** containing expected sample sizes for the trial design corresponding to each value in the vector **theta**.
- **errcode** – 0 if no error detected in call to **gsDesign()** or **gsProbability()**, > 0 otherwise.
- **errmsg** – a text string corresponding to **errcode**; for no error, this is "No errors detected."

5.3 The gsDesign Class

The third and final object class is gsDesign. This inherits the class gsProbability and, in addition, has the following members on output from **gsDesign()**: **test.type**, **alpha**, **beta**, **delta**, **n.fix**, **timing**, **tol**. In addition, all elements of the class spendfn are returned in **upper** and **lower**. Most of these variables are as described in the input to **gsDesign()** or in the spendfn or gsProbability class description. The exceptions are as follows:

- **timing** – when this is input as 1, the output is transformed to a vector of equally spaced analyses as follows: $(1, 2, \dots, k)/k$. Otherwise, this should be a vector of length **k**, with increasing values greater than 0 and the greatest value equal to 1. A value of **timing**[2]=0.4 is translated as the second interim analysis being performed after 40% of the total observations planned for the final analysis.
- **delta** – the standardized effect size; if this was input as 0, it is recomputed with the following formula:

$$\delta = \frac{\Phi^{-1}(1 - \alpha) + \Phi^{-1}(1 - \beta)}{\sqrt{N_{fix}}}$$

where $\delta = \text{delta}$, $N_{fix} = \text{n.fix}$, $\alpha = \text{alpha}$, and $\beta = \text{beta}$.

- **n.I** – this is the sample size or information required at each analysis, depending on how **gsDesign()** is called. In addition to the following descriptions, see DETAILED EXAMPLES.
1. If input to **gsDesign()** was **n.fix** = 1 and **delta**=0, this results in returning a sample-size multiplier in **n.I** that can be used to set sample size at interim analyses according to the sample size required for a fixed design without interim analysis. That is, use a formula for a fixed design that matches the type of data you are collecting, and multiply this fixed design sample size by **n.I** to obtain sample sizes at interim and final analyses for the desired group sequential design.

2. Similarly, if you have a fixed design sample size which you wish to modify using `gsDesign()` to obtain an appropriate group sequential design, input `n.fix` as the fixed sample size design and input `delta=0`. In this case `n.I` will return with sample sizes for each analysis in the group sequential design.
- `upper`, `lower` – these are output as class `spendfn`, as described above. All elements of the `spendfn` class are returned when these are generated from a call to `gsDesign()`.

6 Formatted Output

`gsDesign()` returns an object of the class `gsDesign`. `gsProbability()` returns an object of class `gsDesign` if a `gsDesign` class object is passed in or of class `gsProbability`, if not. The standard R functions `print()`, and `plot()` are extended to work for both the `gsDesign` and the `gsProbability` classes. There are two plot types, each of which is available for both classes: boundary plots and boundary crossing probability plots. Note also that `summary()` prints a brief summary of either object type if you need a reminder of what is in a class. The `print()` function for each object class has a single argument and is implemented through the functions `print.gsDesign()` and `print.gsProbability()`. The `plot()` functions for `gsDesign` and `gsProbability` objects have a second argument which specifies which type of plot is to be made; these functions are implemented with the functions `plot.gsDesign()` and `plot.gsProbability()`, respectively. Further details of the `print()`, and `plot()` functions are provided in DETAILED EXAMPLES below. Since the plot functions have multiple arguments, the use of these arguments is formally specified here.

```
plot.gsDesign(x,plottype=1, main="Default", ylab="Default", xlab="Default")
plot.gsProbability(x,plottype=2, xval="Default", main="Default",
  ylab="Default", xlab="Default")
```

- `x` – for `plot.gsDesign()`, this is an object in the `gsDesign` class; for `plot.gsProbability()`, this is an object in the `gsProbability` class.
- `plottype` – a value of 1 indicates a plot of the boundaries to be used for stopping the trial. A value of 2 produces a boundary crossing probability plot across values of `theta` specified in `x`.
- `xval` – only in the plot function for the `gsProbability` class and only effective when `plottype=2`. This can input a vector of the same length as `x$theta` for plotting on the x-axis instead of `x$theta` (i.e., a different parameterization). The default is that `x$theta` will be used as the x-value.
- `main` – Plot title. "Default" is replaced appropriately depending on `plottype`.
- `xlab` – x-axis label. "Default" is replaced appropriately depending on `plottype`.
- `ylab` – y-axis label. "Default" is replaced appropriately depending on `plottype`.

7 Spending Functions

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 to some. It is also possible for users can to write their own spending functions to pass directly to `gsDesign()`. Spending functions available and the syntax for writing a function with a new spending function are documented here. For users needing more background in spending

functions, basic material is included in STATISTICAL METHODS. We begin here with simple examples of how to apply a 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.

7.1 Spending Function Basics

As noted previously, the default spending function for `gsDesign()` is `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. To change these to -3 (less conservative than an O'Brien-Fleming bound) and 1 (an aggressive Pocock-like bound), respectively, requires only the following:

```
> gsDesign(sfupar=-3,sflpar=1)
```

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) is specified as follows:

```
> gsDesign(sfu=sfpower,sfupar=3,sfl=sfpower,sflpar=0.75)
```

O'Brien-Fleming, Pocock and Wang-Tsiatis ($\Delta=0.25$; between Pocock and O'Brien-Fleming) bounds can be obtained (only for `test.type=1` or `2`) as follows:

```
> gsDesign(test.type=2,sfu="OF")
> gsDesign(test.type=2,sfu="Pocock")
> gsDesign(test.type=2,sfu="WT",sfupar=0.25)
```

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 [2]. 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 3 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. Similarly, Jennison and Turnbull [3], suggest that the Kim-DeMets spending function is flexible enough to suit most purposes. Try this same plot using `sfpower()` instead of `sfHSD()`; use parameter values 3, 2 and 0.75 in the third argument replace the values -4, -2, and 1 in the code below. This will allow a comparison between the Hwang-Shih-DeCani and Kim-DeMets families. The code below yields the plot in the figure below (note the typesetting of Greek characters!):

```
plot(0:100/100,sfHSD(.025,0:100/100,-4)$spend,type="l",lwd=2,
     xlab="Proportion of information",
     ylab=expression(paste("Cumulative ",alpha,"-spending")),
     main="Hwang-Shih-DeCani Spending Functions")
lines(0:100/100,sfHSD(.025,0:100/100,-2)$spend,lty=2,lwd=2)
lines(0:100/100,sfHSD(.025,0:100/100,1)$spend,lty=3,lwd=2)
legend(x=c(.0,.27),y=.025*c(.8,1),lty=1:3,lwd=2,
```

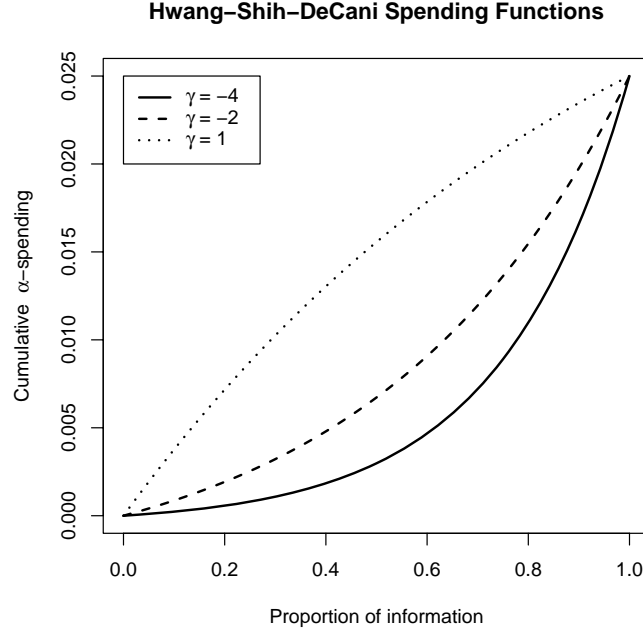


Figure 1: Hwang-Shih-DeCani spending function example

```

legend=c(expression(paste(gamma," = -4")),
          expression(paste(gamma," = -2")),
          expression(paste(gamma," = 1")))

```

Users satisfied with these previously published and standard options may stop here!

7.2 Spending Function Details

Except for the "OF", "Pocock" and "WT" examples just given, a spending function passed to `gsDesign()` 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, $0 \leq t_1 < t_2 < \dots < t_m \leq 1$. 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 will be 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 THE `spendfn` CLASS.

Table 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: **sfexp()** and **sflogistic()**. These spending functions are discussed further in 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 2-parameter spending functions that can provide flexibility not available from one-parameter families. The **sfexp()** spending function is included here as it provides an excellent approximation of an O'Brien-Fleming design as follows:

```
> gsDesign(test.type=2,sfu=sfexp,sfupar=0.75)
```

See also subsections below and online documentation of spending functions. Mathematical background is in STATISTICAL METHODS.

Table 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 |
| sfLD0F (none) | O'Brien-Fleming approximation | $2 \left(1 - \Phi \left(\frac{\Phi^{-1}(\alpha/2)}{\sqrt{t}} \right) \right)$ | None |
| sfpoints (p₁,p₂,...,p_K) | Pointwise specification | Specified points $0 < p_1 \dots < p_K = 1$ | Cumulative proportion of total boundary crossing probability for each analysis |
| sfexp (nu) | Exponential | $\alpha t^{-\nu}$ | $(0, 10]$ Recommend $\nu < 1$ 0.75 =O'Brien-Fleming-like |
| sflogistic (a,b) | Logistic | $\alpha \frac{e^a \left(\frac{t}{1-t} \right)^b}{1 + e^a \left(\frac{t}{1-t} \right)^b}$ | $b > 0$ |
| "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 $\Delta = 1/2$. |
| "OF" (none) | O'Brien-Fleming bounds | | This is a special case of WT with $\Delta = 0$. |

7.3 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 [6], the GUSTO V trial [7] and three ongoing trials at Merck.

The approach allows fitting spending functions by solving a linear system of 2 equations in 2 unknowns. A 2-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 :

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

Fix two points of the spending function $0 < t_0 < t_1 < 1$ to have spending function values specified by $u_0 \times \alpha$ and $u_1 \times \alpha$, respectively, where $0 < u_0 < u_1 < 1$. Equation (1) 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 $t_0 < t_1, u_0 < u_1$.

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 [7]. 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 (1) we obtain the logistic spending function family:

$$\alpha(t; a, b) = \alpha \times \text{logit}^{-1}(\log(c) + b \times \text{logit}(u)) \quad (2)$$

$$= \alpha \frac{c \left(\frac{t}{1-t} \right)^b}{1 + c \left(\frac{t}{1-t} \right)^b} \quad (3)$$

The logistic spending function is implemented in **sflogistic()**. Functions are also available replacing $F()$ with the cumulative distribution functions for the standard normal distribution (**sfnorm()**), the central t-distribution (**sfTDist()**), and the Cauchy distribution (**sfcauchy()**). Of these, **sfnorm()** is fairly similar to **sflogistic()**. On the other hand, **sfcauchy()** can behave quite differently. The function **sfTDist()** provides intermediary spending functions bounded by **sfnorm()** and **sfcauchy()**; it requires an additional parameter, the degrees of freedom. See online help for more complete documentation, particularly for **sfTDidst()**. Following is an example that plots several of these spending functions that fit through the same two points ($t_1=0.25, t_2=0.5, u_1=0.1, u_2=0.2$) but behave differently for $t > 1/2$.

```
plotsf<-function(alpha,t,param)
{ plot(t,sfcauchy(.025,t,param)$spend,lwd=2,
  xlab="Proportion of enrollment",
```

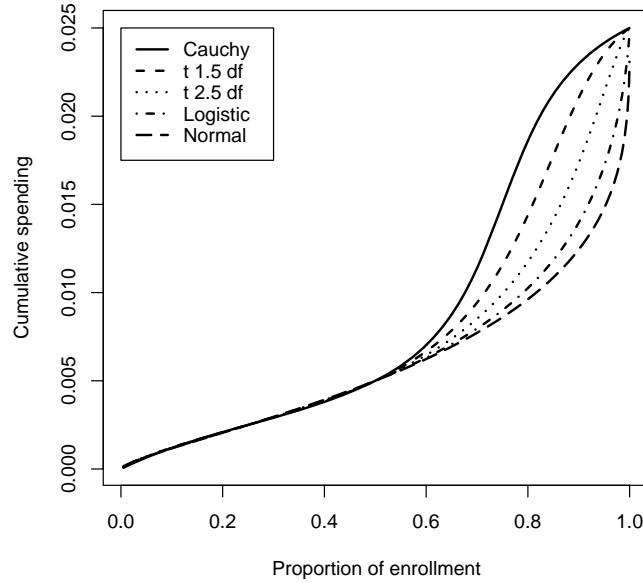


Figure 2: Example fitting 2- and 3-parameter spending functions

```

ylab="Cumulative spending",type="l")
lines(t,sflogistic(.025,t,param)$spend,lty=4,lwd=2)
lines(t,sfnorm(.025,t,param)$spend,lty=5,lwd=2)
lines(t,sfTDist(.025,t,c(param,1.5))$spend,lty=2,lwd=2)
lines(t,sfTDist(.025,t,c(param,2.5))$spend,lty=3,lwd=2)
legend(x=c(.0,.3),y=.025*c(.7,1),lty=1:5,lwd=2,
       legend=c("Cauchy","t 1.5 df","t 2.5 df","Logistic","Normal"))
}
t<-1:199/200
t<-c(t,.9975,.99875,.9995,.99975)
param<-c(.25,.5,.1,.2)
plotsf(.025,t,param)

```

7.4 Optimized Spending Functions

The following 2 examples demonstrate some of the flexibility and research possibilities for the gs-Design 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 [1] which minimized expected value of the square of sample size over a family of spending functions and a prior distribution.

Example 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 ($\Delta=0$) and Pocock ($\Delta=0.5$) 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 \mathbf{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.

```
# Wang-Tsiatis comparison with a one-parameter spending function
WTapprox<-function(x,alpha=0.025,beta=.1,k=3,sf=sfHSD,Delta=.25,test.type=2)
{
  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 2-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.

```
> nu<-nlminb(.67,WTapprox,lower=0,sf=sfexp,k=4,Delta=0,test.type=2)$par
> nu
[1] 0.7562779
```

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.

Example 2 *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 `sf1par` are specified in 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.

```
# 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].
# val is the power of the sample size for which expected
#   values are computed.
# theta is a vector for which expected values are to be computed.
# thetawgt is 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
  x<-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
  x<-gsProbability(theta=theta,d=x)
  # compute sample sizes to the val power
  n<-x$n.I^val
  # compute expected values
  en<-n%*%(x$upper$prob+x$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 [1]. The code above is more general than in [1], 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(.025)+qnorm(.1))
# use normal distribution to get weights
x<-norpts(mu=delta,sigma=delta/2)
x<-nlminb(start=c(.7,-.8),enasfpar,theta=x$z,timing=1,thetawgt=x$wgts,
  val=2,k=4,test.type=5,tol=0.000000001)
x$message
y<-gsDesign(k=4,test.type=5,sfupar=x$par[1],sflpar=x$par[2])
y

```

7.5 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 will be 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=.3,
    "b-dist example spending function parameter must be of length 2"))
  if (!is.numeric(param)) return(gsReturnError(x,errcode=.1,
    errmsg="Beta distribution spending function parameter must be numeric"))
  if (param[1]<=0.) return(gsReturnError(x,errcode=.12,
    errmsg="1st Beta distribution spending function parameter must be > 0.))
  if (param[2]<=0.) return(gsReturnError(x,errcode=.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 $\rho=2$. 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(.5,.5))$spend,lty=3,lwd=2)
lines(0:100/100,sfbdist(1,0:100/100,c(.6,2))$spend,lty=4,lwd=2)
legend(x=c(.65,1),y=1*c(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"))
```

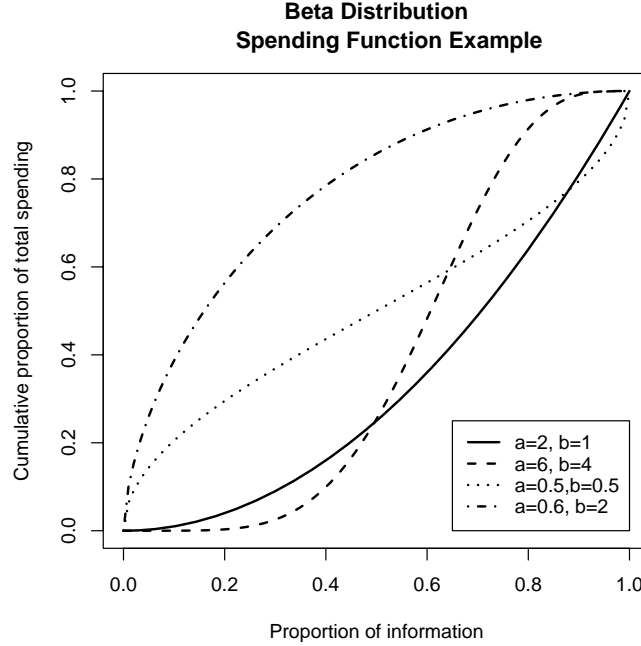


Figure 3: Example plotting user-written beta-distribution spending function

8 Detailed Examples

Below are six examples, each followed by output generated to demonstrate how to use the functions in the package. Example 1 shows the calculation of information ratios and boundaries based on default values for `gsDesign()`. This also demonstrates the difference in sample size when assuming binding versus non-binding (the default) lower bounds. Example 2 demonstrates two-sided testing and user-specified spending; commands demonstrating O'Brien-Fleming, Pocock and Wang-Tsiatis bounds are also shown. Example 3 demonstrates use of the logistic spending function. There are also further comments there on the other 2-parameter spending functions as well as the 3-parameter t-distribution spending function. Example 4 shows how to use `gsProbability()` to calculate boundary crossing probabilities. Example 5 demonstrates how to design a non-inferiority study. Example 6 demonstrates a non-inferiority study that also evaluates superiority.

Example 1: Default input and standard spending functions

For this example, we begin by noting some defaults for `gsDesign()`, and continue with the default call and its associated standard print and plot output. Next, we show the structure of information returned by `gsDesign()`. Since the defaults provide sample size ratios for a group sequential design compared to a design with no interim analysis, we demonstrate how to generate sample sizes for a binomial trial with a binomial endpoint. Finally, we demonstrate some standard spending functions and how to set their corresponding parameters.

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

1. Overall Type I error $\alpha=0.025$ (one-sided)
2. Overall Type II error $\beta=0.1$ (Power=90%)

3. 2 interim analyses plus the final analysis ($k=3$)
4. Asymmetric boundaries which means we may stop the trial for futility or superiority at an interim analysis.
5. Beta-spending is used to set the lower stopping boundary. This means that the spending function controls the incremental amount of Type II at each analysis.
6. 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 is the case, regulators often demand that the lower bounds be ignored when computing Type I error.
7. Hwang-Shih-DeCani spending functions with $\gamma = -4$ for the upper bound and $\gamma = -2$ for the lower bound. This provides a conservative, O'Brien-Fleming like superiority bound and a less conservative lower bound.

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. Next, a boundary plot is generated using `plot(x)`. A power plot is generated with the statement `plot(x,plottype=2)`. The solid lines in this plot are, in ascending order, the cumulative power of the design 1st and 2nd interims and final analysis, respectively, for different values of θ/δ , where δ is the standardized treatment effect for which the trial is powered and θ is the true/underlying standardized treatment effect. The dashed lines, in descending order, are one minus the probability of crossing the lower boundary for the 1st and 2nd interims, respectively.

```
>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 non-group sequential design

Boundary crossing probabilities and expected sample size assuming any cross
stops the trial

Upper boundary (power or Type I Error)
      Analysis
      Theta  1      2      3      Total  E{N}
```

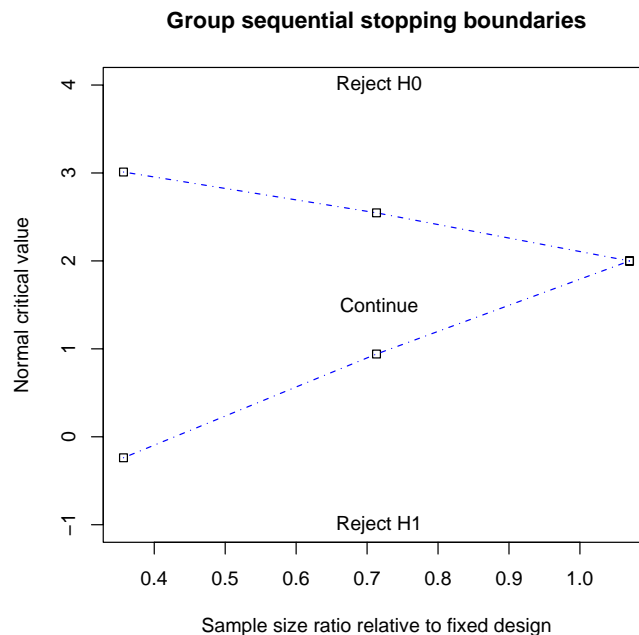


Figure 4: Default plot for gsDesign object from example 1

```
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 |

```
> plot(x)
> plot(x,plottype=2)
```

Above we have seen standard output for `gsDesign()`. Now we look at how to access individual items of information about what is returned from `gsDesign()`. First, we use `summary(x)` to list the elements of `x`. To view an individual element of `x`, we provide the example `x$delta`. See STATISTICAL METHODS for an explanation of this value of `x$delta`. Other elements of `x` can be accessed in the same way. 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. The command `summary x$upper` shows what these variables are. The upper boundary is shown with the command `x$upper$bound`.

```
> summary(x)
      Length Class  Mode
k          1  -none-  numeric
test.type  1  -none-  numeric
alpha      1  -none-  numeric
```

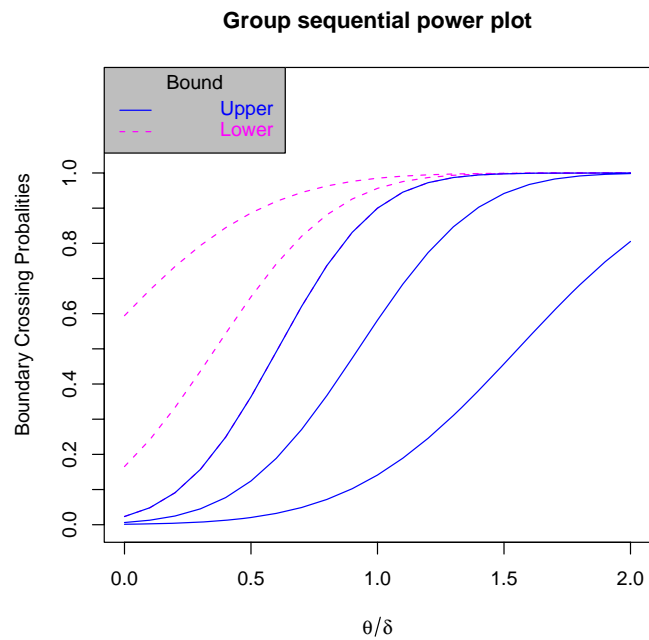


Figure 5: Power plot (plottype=2) for gsDesign object from example 1

```

beta      1      -none-  numeric
astar     1      -none-  numeric
delta     1      -none-  numeric
n.fix     1      -none-  numeric
timing     3      -none-  numeric
tol       1      -none-  numeric
r         1      -none-  numeric
errcode   1      -none-  numeric
errmsg    1      -none-  character
upper     9      spendfn list
lower     9      spendfn list
n.I       3      -none-  numeric
theta     2      -none-  numeric
falseposnb 3      -none-  numeric
en        2      -none-  numeric
> x$delta
[1] 3.241516
> summary(x$upper)
      Length Class  Mode
name     1      -none- character
param    1      -none-  numeric
parname  1      -none- character
sf       1      -none-  function
spend    3      -none-  numeric
bound    3      -none-  numeric

```

```

prob      6      -none-  numeric
errcode  1      -none-  numeric
errmsg   1      -none-  character
> x$upper$bound
[1] 3.010739 2.546531 1.999226

```

Now suppose you wish to design a trial for a binomial outcome 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. A trial with no interim analysis has a sample size of 957 per arm or 1904 based on calculations from nQuery Advisor for a Chi-square test without continuity correction. To get a sample size for the above design, we compute interim and final sample sizes (both arms combined) as follows:

```

> ceiling(2*957*x$n.I)
[1] 683 1366 2048

```

That is, we use `x$n.I` to adjust a fixed sample size design and obtain sample sizes for testing at the interim and final analyses. This method of calculating `x$n.I` is done automatically with the default input value of `n.fix=1`. The following gets this specific trial design with the original call to `gsDesign()`, now with a calculated standardized effect size of $\delta = 0.0741$:

```

> gsDesign(n.fix=2*957)

```

If it were acceptable to use a logrank test for the above design with a fixed follow-up period per patient, nQuery returns a sample size of 645 per arm for a fixed design. In this case, the following sample sizes could be used at the interim and final analyses (the ceiling function is used to round up):

```

> ceiling(gsDesign(n.fix=2*645)$n.I)
[1] 461 921 1381

```

If, in addition, it were acceptable to assume the lower bound was binding, the sample size would be:

```

> ceiling(gsDesign(n.fix=2*645,test.type=3)$n.I)
[1] 451 902 1353

```

Before we proceed to example 2, we consider some simple alternatives to the standard spending function parameters. 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 use a Kim-DeMets (power) spending function for both lower and upper bounds with parameters 2 and 3, respectively. Then we compare bounds with the above design.

```

> 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] 0.3989132 1.3302944 2.0637399
> xKD$lower$bound
[1] -0.3497491 0.9822541 2.0087052

```

Example 2: 2-sided testing, including pointwise spending, O'Brien-Fleming, Pocock and Wang-Tsiatis designs

For this example we consider a 2-sided test with user-specified spending and 5 analyses with unequal spacing. We again assume the binomial example with fixed n per arm of 957. Note the difference in labeling inside the plot due to the 2-sided nature of testing. Note also that spending is printed as one-sided, and that only the upper spending function is needed/used. The cumulative spending at each analysis

```

> # Cumulative proportion of spending planned at each analysis
> p=c(.05,.1,.15,.2,1)
> # Cumulative spending intended at each analysis (for illustration)
> p*0.025
[1] 0.00125 0.00250 0.00375 0.00500 0.02500
> # Incremental spending intended at each analysis
> # for comparison to spend column in output
> (p-c(0,p[0:4]))*0.025
[1] 0.00125 0.00125 0.00125 0.00125 0.02000
> x<-gsDesign(k=5,test.type=2,n.fix=1904,timing=c(.1,.25,.4,.6),
+ sfu=sfpoints,sfupar=p)
> x
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.

```

| Analysis | N | Z | Nominal p | Spend |
|----------|------|------|-----------|--------|
| 1 | 196 | 3.02 | 0.0013 | 0.0013 |
| 2 | 488 | 2.99 | 0.0014 | 0.0013 |
| 3 | 781 | 2.93 | 0.0017 | 0.0012 |
| 4 | 1171 | 2.90 | 0.0019 | 0.0013 |
| 5 | 1952 | 2.01 | 0.0222 | 0.0200 |
| Total | | | | 0.0250 |

```

++ alpha spending:  User-specified spending function with
Points = 0.05 0.1 0.15 0.2 1

```

Boundary crossing probabilities and expected sample size assuming any cross stops the trial

Upper boundary (power or Type I Error)

| Theta | 1 | 2 | 3 | 4 | 5 | Total | E{N} |
|--------|--------|--------|--------|--------|--------|-------|--------|
| 0.0000 | 0.0013 | 0.0013 | 0.0013 | 0.0013 | 0.0200 | 0.025 | 1938.4 |
| 0.0743 | 0.0235 | 0.0758 | 0.1218 | 0.1760 | 0.5029 | 0.900 | 1519.1 |

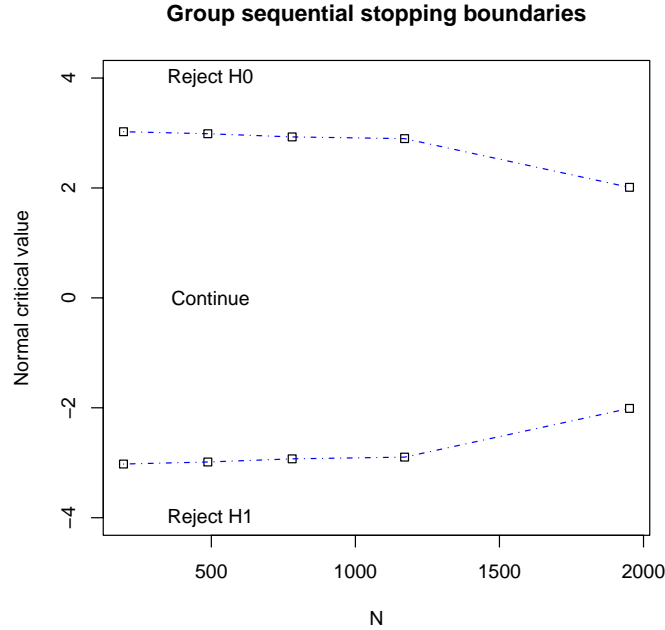


Figure 6: Boundary plot for example 2

```
Lower boundary (futility or Type II Error)
Analysis
  Theta 1      2      3      4      5  Total
0.0000 0.0013 0.0013 0.0013 0.0013 0.02 0.025
0.0743 0.0000 0.0000 0.0000 0.0000 0.00 0.000
> plot(x)
```

O'Brien-Fleming, Pocock or Wang-Tsiatis are normally used with equally-spaced analyses. O'Brien-Fleming, Pocock or Wang-Tsiatis (parameter of 0.4) bounds for equally space analyses are generated as follows:

```
xOF<-gsDesign(k=5,test.type=2,n.fix=1904,sfu="OF")
xPk<-gsDesign(k=5,test.type=2,n.fix=1904,sfu="Pocock")
xWT<-gsDesign(k=5,test.type=2,n.fix=1904,sfu="WT",sfupar=.4)
```

Once you have generated these designs, examine the upper bounds as in example 1. Also, look at the spending by looking at, for example, `xOF$upper$spend`.

Example 3: Logistic spending function

Assume we would like the cumulative spending at 10% of enrollment to be 0.00125 (5% of total spending) and at 60% of enrollment to be 0.005 (20% of total spending) so that alpha-spending of 0.02 is available for the final analysis; see `sfupar` in the following to find these numbers. This is the 4-parameter specification of a logistic spending function (`sflogistic()`), or the other 2-parameter spending functions `sfnorm()` and `sfcauchy()`. In each case, the four-parameter specification is translated to the 2 essential parameters, but allows the specification to be done simply.


```
> gsDesign(k=5,timing=c(.1,.25,.4,.6),test.type=2,n.fix=1904,
+ sfu=sflogistic,sfupar=c(.1,.6,.05,.2))
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.
```

```
Analysis   N    Z   Nominal p Spend
      1  195 3.02    0.0013 0.0013
      2  488 3.04    0.0012 0.0011
      3  780 2.99    0.0014 0.0010
      4 1170 2.83    0.0023 0.0017
      5 1949 2.01    0.0223 0.0200
      Total 0.0250
++ alpha spending: Logistic spending function with
a b = -1.629033 0.5986671
```

Boundary crossing probabilities and expected sample size assuming any cross stops the trial

```
Upper boundary (power or Type I Error)
      Analysis
Theta  1      2      3      4      5   Total   E{N}
0.0000 0.0013 0.0011 0.0010 0.0017 0.020 0.025 1936.1
0.0743 0.0235 0.0686 0.1123 0.2077 0.488 0.900 1514.0
```

```
Lower boundary (futility or Type II Error)
      Analysis
Theta  1      2      3      4      5   Total
0.0000 0.0013 0.0011 0.001 0.0017 0.02 0.025
0.0743 0.0000 0.0000 0.000 0.0000 0.00 0.000
```

The same output can be obtained using the 2-parameter specification of the logistic spending function as follows (with values for a and b from above in param):

```
> y<-gsDesign(k=5,timing=c(.1,.25,.4,.6),test.type=2,n.fix=1904,
+ sfu=sflogistic,sfupar=c(-1.629033,0.5986671))
```

To verify the initial spend is 0.00125 (since it was rounded to 0.0013 above), we examine the appropriate element of y just computed:

```
> y$upper$spend[1]
[1] 0.00125
```

Example 4: gsProbability()

We reconsider example 1 and obtain the properties for the design for a larger set of θ values than in the standard printout for `gsDesign()`. The standard plot design for a `gsProbability` object is the power plot shown in example 1. The boundary plot is obtainable below using the command `plot(y,plottype=1)`.

```
> x<-gsDesign()
```

```

> y<-gsProbability(theta=x$delta*seq(0,2,.25),d=x)
> y
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 non-group sequential design

```

Boundary crossing probabilities and expected sample size assuming 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 |
| 0.8104 | 0.0058 | 0.0279 | 0.0872 | 0.1209 | 0.7523 |
| 1.6208 | 0.0205 | 0.1038 | 0.2393 | 0.3636 | 0.8520 |
| 2.4311 | 0.0595 | 0.2579 | 0.3636 | 0.6810 | 0.8668 |
| 3.2415 | 0.1412 | 0.4403 | 0.3185 | 0.9000 | 0.7913 |
| 4.0519 | 0.2773 | 0.5353 | 0.1684 | 0.9810 | 0.6765 |
| 4.8623 | 0.4574 | 0.4844 | 0.0559 | 0.9976 | 0.5701 |
| 5.6727 | 0.6469 | 0.3410 | 0.0119 | 0.9998 | 0.4868 |
| 6.4830 | 0.8053 | 0.1930 | 0.0016 | 1.0000 | 0.4266 |

Lower boundary (futility or Type II Error)

| Analysis | | | | |
|----------|--------|--------|--------|--------|
| Theta | 1 | 2 | 3 | Total |
| 0.0000 | 0.4057 | 0.4290 | 0.1420 | 0.9767 |
| 0.8104 | 0.2349 | 0.3812 | 0.2630 | 0.8791 |
| 1.6208 | 0.1138 | 0.2385 | 0.2841 | 0.6364 |
| 2.4311 | 0.0455 | 0.1017 | 0.1718 | 0.3190 |
| 3.2415 | 0.0148 | 0.0289 | 0.0563 | 0.1000 |
| 4.0519 | 0.0039 | 0.0054 | 0.0097 | 0.0190 |
| 4.8623 | 0.0008 | 0.0006 | 0.0009 | 0.0024 |
| 5.6727 | 0.0001 | 0.0001 | 0.0000 | 0.0002 |
| 6.4830 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

Example 5: Non-inferiority testing

We consider a trial examining a new drug that is more convenient to administer than an approved control. There is no expectation of a substantially improved response with the new drug. While the

new drug is may be a little better or worse than control, there is some suggestion that the new drug may not be as efficacious as control. Rather than powering the trial to show non-inferiority when the new drug is slightly worse than control, the strategy is taken to stop the trial early for futility if there is a ‘substantial’ trend towards the new drug being inferior. The control drug has provided a (binomial) response rate of 67.7% in a past trial and regulators have agreed with a non-inferiority margin of 7%. Let the underlying event rate in the control and experimental groups be denoted by p_C and p_E , respectively. Let $\delta = 0.07$ represent the noninferiority margin. There is no desire to stop the trial early to establish noninferiority. That is, this is a one-sided testing problem for interim analyses. We let $H_0: p_C - p_A \leq 0$ and test against the alternative $H_1: p_C - p_A \geq \delta$, only stopping early if H_0 can be rejected. We must have 97.5% power to reject H_0 when H_1 is true ($\beta = 0.025$) and can have a 10% chance of rejecting H_0 when H_0 is true ($\alpha = 0.1$). In this case, an aggressive stopping boundary is desirable to stop the trial 40% of the way through enrollment if the experimental drug is, in fact, not as efficacious as control. Using PASS 2005, this requires 938 patients per treatment group (1876 total) for a trial with no interim analysis. When used with `gsDesign()`, this yields a sample size of 2226 for the trial, the same as EAST 5.1. This design requires less than approximately a 4.5% difference in event rates at the interim analysis to continue and a final difference of no more than approximately 3.3% to achieve non-inferiority. These differences were carefully evaluated in choosing the appropriate value of `gamma` for the spending function.

```
> gsDesign(k=2,alpha=.1,beta=.025,n.fix=1876,test.type=1,sfupar=3,timing=.4)
One-sided group sequential design with 97.5 % power and 10 % Type I Error.
```

```

Analysis   N   Z   Nominal p   Spend
      1  891 1.45    0.0735 0.0735
      2 2226 1.68    0.0468 0.0265
      Total                                0.1000
++ alpha spending:  Hwang-Shih-DeCani spending function with gamma = 3
Boundary crossing probabilities and expected sample size assuming any cross
stops the trial
```

```
Upper boundary (power or Type I Error)
```

```

Analysis
Theta   1       2       Total   E{N}
0.0000 0.0735 0.0265 0.100 2127.7
0.0748 0.7832 0.1918 0.975 1179.8
```

Example 6: Non-inferiority and superiority testing in the same trial

We consider a safety trial where a drug is given chronically to patients who are expected to have a 3.5% annual risk (exponential parameter $\lambda_0 = 0.035627$) of developing cardiovascular disease (CVD) and we wish to rule out an elevated risk that would be indicated by a hazard ratio of 1.2 ($\lambda_1 = 0.042753$). The desire is that if there is a hazard ratio of 1.2 that there is at most a 2.5% chance of demonstrating non-inferiority. On the other hand, if the true hazard ratio is 1 (no excess risk), we wish to have a 90% probability of showing ‘no difference.’ In hypothesis testing terms, the role of the null hypothesis (no difference) and alternative hypothesis (20% increased hazard ratio) have been reversed. We will label the hypotheses as before, but the error levels will be reversed to satisfy the above. We let the null hypothesis of no difference be denoted by $H_0: \log(\lambda_1/\lambda_0) = 0$ and the alternate hypothesis will be denoted by $H_1: \log(\lambda_1/\lambda_0) = \log(1.2)$. To achieve the desired performance, Type I error is set to 10% and Type II error is set to 2.5%. Assuming a trial is to be enrolled in a 2-year period and the dropout rate is 15% ($\lambda_D = 0.162519$), nQuery Advisor was used to find that a fixed design with no interim analysis requires a sample size of 6,519 per treatment

group to obtain 1,264 total events in 4.5 years under the alternate hypothesis. Note that under the null hypothesis, the overall event rate would be lower and it would take longer to obtain the number of events required. We will use `n.fix=1,264` below to adjust this fixed design to a group sequential design and obtain the number of events required at each analysis of a group sequential trial. In addition, we consider the possibility that the experimental drug may actually reduce cardiovascular risk. For a fixed design, superiority testing may be performed following noninferiority testing. For a group sequential trial, the situation is slightly more complex; this is not a scenario that can be dealt with in EAST 5.1.

Let $H_0: \theta = 0$ and $H_1: \theta \neq 0$ where θ indicates the underlying difference between two treatment groups. We wish to show that a new treatment is non-inferior compared to control. That is, for some $\delta > 0$, under $H_{1A}: \theta = \delta$ we wish to have 97.5% power to reject $H_0: \theta = 0$ (i.e., `beta=0.025` to yield a 2.5% chance of accepting non-inferiority when in fact the underlying effect for the experimental group is higher by δ). On the other hand, under $H_0: \theta = 0$ we are willing to have a 10% chance of rejecting $H_0: \theta = 0$ in favor of $H_{1A}: \theta = \delta$ (`alpha=0.10` to yield 90% power to show non-inferiority). We have a slightly asymmetric test for superiority in that we would like to reject $H_0: \theta=0$ in favor of $H_{1B}: \theta = -\delta$ at the 2.5% (one-sided) level (`astar=0.025`) to control Type I error in that direction. Thus, the trial could stop early if θ is substantially different from 0 in either direction. A Hwang-Shih-DeCani spending function is used for each bound. The cutoffs are aggressive for detecting excess risk of CVD in the experimental group (`sfupar=0`) and relatively conservative to demonstrate that the experimental group has a lower CVD rate (`sfupar=-3`). The former is not quite as aggressive as a Pocock boundary and the latter is not quite as conservative as an O'Brien-Fleming bound. Thus, the bounds are asymmetric due both to this difference in the spending function parameters and due to the different levels of the test in each direction. The appropriate confidence interval approach for this design is probably a stage-wise ordering approach (see Jennison & Turnbull [3], sections 8.4 and 8.5). This will give the tightest intervals at the end of the trial and will not result in conflicts between the confidence intervals and the testing approach just outlined.

See output below for a design with 4 equally spaced interims. With `n.fix=1264` we find that $\delta = 0.912$. Interestingly, there is 92% power to find superiority under $H_{1B}: \theta = -\delta$, so the trial is well-powered for superiority. The design requires an increase from 1264 events to 1417 in order to maintain the desired error rates with the aggressive stopping rule for excess risk. Making the upper bound less aggressive may be unethical in that the trial would not be stopped if a substantial excess risk were observed. Note that `test.type=5` indicates that all bounds are binding and both upper and lower bound error spending is under the null hypothesis. The upper bound is a futility bound for showing non-inferiority. If it is never crossed, the trial establishes non-inferiority. The lower bound is the superiority bound.

The plot below shows the cumulative probabilities of stopping at each analysis for different values of θ . The solid lines go from the probability of stopping at the first interim analysis to the highest line which shows the probability of an inferiority finding at any analysis. The dashed lines provide 1 minus the cumulative probability of stopping for superiority for different values of θ for each analysis. Note that the expected sample size required to come to a conclusion if the experimental regiment is truly inferior to control ($H_{1A}: \theta = \delta$) is 653 events compared to 1264 for a fixed design. In addition, there is only a 6.8% chance of continuing the trial to the final analysis. This is comprised of a 2.5% of continuing the trial to the final analysis and falsely establishing noninferiority plus a 4.27% of continuing to the trial to the final analysis to establish inferiority. If experimental therapy is truly superior, the expected number of events required to demonstrate a difference is 950. If there is truly no difference the trial usually goes to the end (about 91% of the time; this is seen by the last result below) and finds no difference (87.5% of the time; this is the sum of the Type I error in both directions).

Evaluation of the cutoffs has not been performed as in example 5. This exercise would be worthwhile to verify that the cutoffs of the design are appropriate.

```
> x<-gsDesign(test.type=5,k=5,alpha=.1,beta=.025,astar=.025,sflpar=-3,
sfupar=0,n.fix=1264)
> y<-gsProbability(d=x,theta=c(-x$delta,0,x$delta))
> y
Asymmetric two-sided group sequential design with 97.5 % power
and 10 % Type I Error.
Spending computations assume trial stops if a bound is crossed.
```

```

      ----Lower bounds----  ----Upper bounds-----
Analysis   N    Z   Nominal p Spend+   Z   Nominal p Spend++
      1  284 -3.07   0.0011 0.0011  2.05   0.0200   0.02
      2  567 -2.84   0.0022 0.0020  1.91   0.0278   0.02
      3  850 -2.60   0.0047 0.0036  1.79   0.0368   0.02
      4 1133 -2.34   0.0097 0.0065  1.68   0.0465   0.02
      5 1417 -2.06   0.0197 0.0119  1.58   0.0568   0.02
      Total                0.0250                                0.1000
+ lower bound spending (under H0):  Hwang-Shih-DeCani spending function
with gamma = -3
++ alpha spending:  Hwang-Shih-DeCani spending function with gamma = 0
```

Boundary crossing probabilities and expected sample size assuming any cross stops the trial

Upper boundary (power or Type I Error)

```

      Analysis
      Theta    1    2    3    4    5  Total  E{N}
-0.0912 0.0002 0.000 0.0000 0.0000 0.0000 0.0002  950.0
 0.0000 0.0200 0.020 0.0200 0.0200 0.0200 0.1000 1352.8
 0.0912 0.3018 0.325 0.2048 0.1007 0.0427 0.9750  653.6
```

Lower boundary (futility or Type II Error)

```

      Analysis
      Theta    1    2    3    4    5  Total
-0.0912 0.0625 0.1988 0.2796 0.2396 0.1401 0.9207
 0.0000 0.0011 0.0020 0.0036 0.0065 0.0119 0.0250
 0.0912 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

```
> plot(gsProbability(k=5,n.I=x$n.I,a=x$lower$bound,b=x$upper$bound,
theta=x$delta*(-25:25)/25))
> # compute probability of continuing to the end of the trial under H0
> 1-sum(y$upper$prob[1:4,2]+y$lower$prob[1:4,2])
[1] 0.9068707
```

;

9 Statistical Methods

We follow notation and methods provided by Jennison and Turnbull [3]. In particular, we consider the asymptotic approximation they refer to as the ‘canonical form.’ Examples of how this is applied to a binomial trial are above in the DETAILED EXAMPLES section. Computational methods follow Chapter 19 of Jennison and Turnbull [3] and are not provided here.

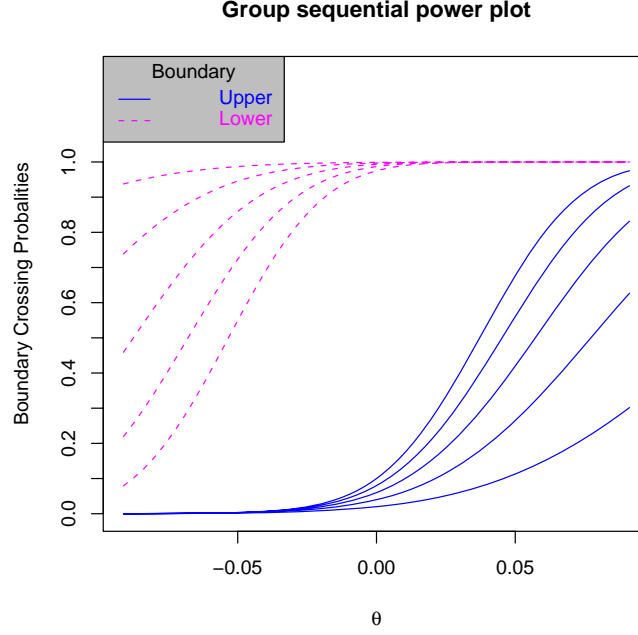


Figure 7: Power plot for non-inferiority design in example 6

Canonical form for asymptotic approximation of group sequential test statistics.

Consider a group sequential trial with $K > 0$ planned analyses, including the final analysis. Let θ represent the single parameter of interest, and assume it can take on any real value. Assume that the distribution of test statistics Z_1, Z_2, \dots, Z_K for the K analyses follows a multivariate normal distribution with

$$E\{Z_i\} = \theta\sqrt{I_i}$$

$$\text{Cov}(Z_j, Z_i) = \sqrt{I_j/I_i},$$

for $1 \leq j \leq i \leq K$. Jennison and Turnbull [3] refer to this as the ‘canonical form’ and show that it is the asymptotic form for a broad variety of group sequential designs. The values $0 < I_1 < I_2 < \dots < I_K$ represent the statistical information available at analyses 1 through K , respectively. In general I_i will be proportional to the number of observations (or events) at interim analysis i , $i = 1, 2, \dots, K$. As an example, assume X_j , $j = 1, 2, \dots$ are independent and normally distributed with mean θ and variance 1. Let n_1, n_2, \dots, n_K be an increasing sequence of positive integers. Let $Z_i = \sum_{j=1}^{n_i} X_j / \sqrt{n_i}$, $i = 1, 2, \dots, K$. Let $I_i = n_i$, $i = 1, 2, \dots, K$. Then the sequence Z_i , $i = 1, 2, \dots, K$ has the canonical distribution specified above. Other software programs such as EAST and the University of Wisconsin software use this distributional assumption as primary tools for group sequential design.

We assume that the primary interest is to test the null hypothesis $H_0: \theta = 0$ against the alternative $H_1: \theta = \delta$ for a fixed $\delta > 0$. 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 $i = 1, 2, \dots, K - 1$, interim cutoffs $l_i < u_i$ are set; final cutoffs $l_K \leq u_K$ are also set. For $i = 1, 2, \dots, K$, the trial is stopped at analysis i to reject H_0 if $l_j < Z_j < u_j$, $j = 1, 2, \dots, i - 1$ 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 , $i = 1, 2, \dots, K$. Thus, $3K$ parameters define a group sequential design: l_i , u_i , and I_i , $i = 1, 2, \dots, K$. Note that if $l_K < u_K$ there is the possibility of completing the trial without rejecting H_0 or H_1 . We will generally restrict $l_K = u_K$ so that one hypothesis is rejected.

Sample size ratio for a group sequential design compared to a fixed design. Consider a trial with a fixed design with power $100(1-\beta)\%$ and level α (1-sided). Denote the sample size as N_{fix} and statistical information for this design as I_{fix} . For a group sequential design as noted above, we will 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

$$r = I_K / I_{fix} = N_K / N_{fix}. \quad (4)$$

This ratio is independent of the θ -value δ for which the trial is powered as long as the information (sample size) available at each analysis increases proportionately with I_{fix} and the boundaries for the group sequential design remain unchanged. Let $\Phi^{-1}()$ denote the inverse of the cumulative standard normal distribution function. In order for the fixed design with level α (1-sided), to have power $100(1-\beta)\%$ to reject $\theta=0$ when in fact $\theta = \delta$, we must have

$$I_{fix} = \left(\frac{\Phi^{-1}(1-\alpha) + \Phi^{-1}(1-\beta)}{\delta} \right)^2. \quad (5)$$

From (4) and (5), if we let $\delta = \Phi^{-1}(1-\alpha) + \Phi^{-1}(1-\beta)$ then $r = I_K$. Because of this relation, this is the value of δ that is used and displayed when information ratios are computed in `gsDesign()`.

Spending functions. For $i = 1, 2, \dots, K$ denote the probability of stopping and rejecting the null hypothesis at analysis i as a function of θ by

$$\alpha_i(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}. \quad (6)$$

The value $\alpha_i(0)$ is commonly referred to as the amount of α (Type I error rate) spent at analysis i , $1 \leq i \leq K$. The total Type I error rate for a trial will be denoted by $\alpha \equiv \sum_{i=1}^K \alpha_i(0)$. We define a continuously increasing spending function $\alpha(t)$, $0 \leq t \leq 1$ with $\alpha(0) = 0$ and $\alpha(1) = \alpha$ that will be used to set $\alpha_i(0)$, for $i = 1, 2, \dots, K$ using the relation

$$\alpha(I_i/I_K) = \sum_{j=1}^i \alpha_j(0). \quad (7)$$

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

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

For one-sided testing (`test.type=1`) and for non-binding lower bounds (`test.type=4` or `6`), (6) is not used for computing Type I error. In these cases, $\alpha_i(\theta)$ is replaced for $i = 1, 2, \dots, K$ by

$$\alpha_i^+(\theta) = P_\theta\{\{Z_i \geq u_i\} \cap_{j=1}^{i-1} \{Z_j < u_j\}\} \quad (8)$$

and a spending function $\alpha^+(t)$, $0 \leq t \leq 1$ is used to set boundaries using the relation

$$\alpha^+(I_i/I_K) = \sum_{j=1}^i \alpha_j^+(0). \quad (9)$$

Next, we consider analogous notation for the lower bound. For $i = 1, 2, \dots, K$ denote

$$\beta_i(\theta) = P_\theta\{\{Z_i \leq l_i\} \cap_{j=1}^{i-1} \{l_j < Z_j < u_j\}\}. \quad (10)$$

The total lower boundary crossing probability in this case will be written as $\beta(\theta) = \sum_{i=1}^K \beta_i(\theta)$. The total Type II error rate for a trial will be denoted by $\beta \equiv \sum_{i=1}^K \beta_i(\delta)$. For a specified value of θ we define an increasing function $\beta(t; \theta)$ on the interval $[0, 1]$ with $\beta(0; \theta) = 0$ and $\beta(1; \theta) = \beta(\theta)$ where $\beta(\theta)$ is the desired total probability for crossing a lower boundary at any analysis when θ is the true parameter value. We now wish to set lower bounds using this spending function to obtain

$$\beta(I_i/I_K; \theta) = \sum_{j=1}^i \beta_j(\theta). \quad (11)$$

There are two options for how to use (11) to set lower bounds. For `test.type=2`, 5 and 6, we set lower boundary values under the null hypothesis by specifying $\beta(t; 0)$, $0 \leq t \leq 1$. For `test.type=3` and 4, we compute lower boundary values under the alternative hypothesis by specifying $\beta(t; \delta)$, $0 \leq t \leq 1$. $\beta(t; \delta)$ will be referred to as the β -spending function and the value $\beta_i(\delta)$ will be referred to as the amount of β (Type II error rate) spent at analysis i , $1 \leq i \leq K$.

We now have four ways of specifying bounds using spending functions. For upper bounds, we can use spending functions $\alpha(t)$ or $\alpha^+(t)$ while for lower spending we can use spending functions $\beta(t; 0)$ or $\beta(t; \theta)$. These are summarized in the following table:

Table 2. Spending functions by `test.type`.

| Upper bound spending function | Lower bound spending function | <code>test.type</code> |
|----------------------------------|----------------------------------|------------------------|
| $\alpha^+(t)$ | None | 1 |
| $\alpha(t)$ | $\beta(t; \delta)$ | 3 |
| $\alpha^+(t)$ | $\beta(t; \delta)$ | 4 |
| $\alpha(t)$ | $\beta(t; 0)$ | 2, 5 |
| $\alpha^+(t)$ | $\beta(t; 0)$ | 6 |

For `test.type=1`, 2 and 5, boundaries can be computed in a single step as outlined by Jennison and Turnbull [3], Chapter 19 just by knowing the proportion of the total planned sampling 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. In these cases, the total sample size is then set to obtain the desired power under the alternative hypothesis using a root finding algorithm.

For `test.type=3` and 4 the sample size and bounds are all set simultaneously using an iterative algorithm. This computation is relatively straightforward, but more complex than the above. This will 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.

References

- [1] Anderson, K. M. 2007. Optimal spending functions for asymmetric group sequential designs. *Biometrical Journal* 49:337–345.

- [2] Hwang, I. K., W. J. Shih, and J. S. De Cani 1990. Group sequential designs using a family of type 1 error probability spending functions. *Statistics in Medicine* 9:1439–1445.
- [3] Jennison, C. and B. W. Turnbull 2000. *Group Sequential Methods with Applications to Clinical Trials*. Boca Raton, FL : Chapman and Hall/CRC.
- [4] Kim, K. and D. L. DeMets 1987. Design and analysis of group sequential tests based on type i error spending rate functions. *Biometrika* 74:149–154.
- [5] Lan, K. K. G. and D. L. DeMets 1983. Discrete sequential boundaries for clinical trials. *Biometrika* 70:659–663.
- [6] The_CAPTURE_Investigators 1997. Randomised placebo-controlled trial of abciximab before and during coronary intervention in refractory unstable angina: the capture study. *Lancet* 349:1429–1435.
- [7] The_GUSTO_V_Investigators 2001. 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. *Lancet* 357:1905–1914.
- [8] Wang, S. K. and A. A. Tsiatis 1987. Approximately optimal one-parameter boundaries for group sequential trials. *Biometrics* 43:193–199.

10 Contacts

Keaven M. Anderson: keaven_anderson@merck.com