# Package 'gsDesign' 

February 3, 2022
Version 3.2.2
Title Group Sequential Design
Description Derives group sequential clinical trial designs and describes their properties. Particular focus on time-to-event, binary, and continuous outcomes. Largely based on methods described in Jennison, Christopher and Turnbull, Bruce W., 2000,
"Group Sequential Methods with Applications to Clinical Trials" ISBN: 0-8493-0316-8.

Depends R ( $>=3.5 .0$ ), ggplot2 ( $>=0.9 .2$ )
License GPL (>=3)
URL https://keaven.github.io/gsDesign/,
https://github.com/keaven/gsDesign
BugReports https://github.com/keaven/gsDesign/issues
Encoding UTF-8
Imports stats, magrittr, rlang, methods, graphics, tools, dplyr, tidyr, xtable

Suggests knitr, scales, rmarkdown, gt, testthat, covr, gridExtra, tibble, utils, mvtnorm

VignetteBuilder knitr
Copyright Copyright 2010, Merck Research Laboratories
RoxygenNote 7.1.2
NeedsCompilation yes
Author Keaven Anderson [aut, cre]
Maintainer Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)
Repository CRAN
Date/Publication 2022-02-02 23:00:02 UTC

## $R$ topics documented:

checkLengths ..... 2
ciBinomial ..... 5
condPower ..... 11
eEvents ..... 17
gsBinomialExact ..... 20
gsBound ..... 25
gsBoundCP ..... 28
gsCP ..... 29
gsDensity ..... 33
gsDesign ..... 36
gsDesign package overview ..... 43
gsProbability ..... 44
hGraph ..... 47
nNormal ..... 49
normalGrid ..... 52
plot.gsDesign ..... 54
print.nSurv ..... 57
print.nSurvival ..... 66
sequentialPValue ..... 71
sfExponential ..... 73
sfHSD ..... 75
sfLDOF ..... 77
sfLinear ..... 79
sfLogistic ..... 82
sfPoints ..... 85
sfPower ..... 87
sfTDist ..... 89
sfTruncated ..... 91
Spending_Function_Overview ..... 94
summary.gsDesign ..... 97
xtable ..... 103
Index ..... 104
checkLengths Utility functions to verify variable properties

## Description

Utility functions to verify an objects's properties including whether it is a scalar or vector, the class, the length, and (if numeric) whether the range of values is on a specified interval. Additionally, the checkLengths function can be used to ensure that all the supplied inputs have equal lengths.
isInteger is similar to is.integer except that isInteger(1) returns TRUE whereas is.integer (1) returns FALSE.
checkScalar is used to verify that the input object is a scalar as well as the other properties specified above.
checkVector is used to verify that the input object is an atomic vector as well as the other properties as defined above.
checkRange is used to check whether the numeric input object's values reside on the specified interval. If any of the values are outside the specified interval, a FALSE is returned.
checkLength is used to check whether all of the supplied inputs have equal lengths.

## Usage

```
checkLengths(..., allowSingle = FALSE)
checkRange(
    x,
    interval = 0:1,
    inclusion = c(TRUE, TRUE),
    varname = deparse(substitute(x)),
    tol = 0
)
checkScalar(x, isType = "numeric", ...)
checkVector(x, isType = "numeric", ..., length = NULL)
isInteger(x)
```


## Arguments

| $\ldots$. | For the checkScalar and checkVector functions, this input represents addi- <br> tional arguments sent directly to the checkRange function. For the <br> checkLengths function, this input represents the arguments to check for equal <br> lengths. |
| :--- | :--- |
| allowSingle | logical flag. If TRUE, arguments that are vectors comprised of a single element <br> are not included in the comparative length test in the checkLengths function. <br> Partial matching on the name of this argument is not performed so you must <br> specify 'allowSingle' in its entirety in the call. |
| any object. |  |


| tol | numeric scalar defining the tolerance to use in testing the intervals of the <br> checkRange function. |
| :--- | :--- |
| isType | character string defining the class that the input object is expected to be. <br> length |
|  | integer specifying the expected length of the object in the case it is a vector. If <br> length=NULL, the default, then no length check is performed. |

## Value

isInteger: Boolean value as checking result Other functions have no return value, called for side effects

## Examples

```
# check whether input is an integer
isInteger(1)
isInteger(1:5)
try(isInteger("abc")) # expect error
# check whether input is an integer scalar
checkScalar(3, "integer")
# check whether input is an integer scalar that resides
# on the interval on [3, 6]. Then test for interval (3, 6].
checkScalar(3, "integer", c(3, 6))
try(checkScalar(3, "integer", c(3, 6), c(FALSE, TRUE))) # expect error
# check whether the input is an atomic vector of class numeric,
# of length 3, and whose value all reside on the interval [1, 10)
x <- c(3, pi, exp(1))
checkVector(x, "numeric", c(1, 10), c(TRUE, FALSE), length = 3)
# do the same but change the expected length; expect error
try(checkVector(x, "numeric", c(1, 10), c(TRUE, FALSE), length = 2))
# create faux function to check input variable
foo <- function(moo) checkVector(moo, "character")
foo(letters)
try(foo(1:5)) # expect error with function and argument name in message
# check for equal lengths of various inputs
checkLengths(1:2, 2:3, 3:4)
try(checkLengths(1, 2, 3, 4:5)) # expect error
# check for equal length inputs but ignore single element vectors
checkLengths(1, 2, 3, 4:5, 7:8, allowSingle = TRUE)
```


## Description

Support is provided for sample size estimation, power, testing, confidence intervals and simulation for fixed sample size trials (that is, not group sequential or adaptive) with two arms and binary outcomes. Both superiority and non-inferiority trials are considered. While all routines default to comparisons of risk-difference, options to base computations on risk-ratio and odds-ratio are also included.
nBinomial() computes sample size or power using the method of Farrington and Manning (1990) for a trial to test the difference between two binomial event rates. The routine can be used for a test of superiority or non-inferiority. For a design that tests for superiority nBinomial() is consistent with the method of Fleiss, Tytun, and Ury (but without the continuity correction) to test for differences between event rates. This routine is consistent with the Hmisc package routines bsamsize and bpower for superiority designs. Vector arguments allow computing sample sizes for multiple scenarios for comparative purposes.
testBinomial() computes a Z- or Chi-square-statistic that compares two binomial event rates using the method of Miettinen and Nurminen (1980). This can be used for superiority or noninferiority testing. Vector arguments allow easy incorporation into simulation routines for fixed, group sequential and adaptive designs.
ciBinomial() computes confidence intervals for 1) the difference between two rates, 2) the riskratio for two rates or 3) the odds-ratio for two rates. This procedure provides inference that is consistent with testBinomial() in that the confidence intervals are produced by inverting the testing procedures in testBinomial(). The Type I error alpha input to ciBinomial is always interpreted as 2-sided.
simBinomial() performs simulations to estimate the power for a Miettinen and Nurminen (1985) test comparing two binomial rates for superiority or non-inferiority. As noted in documentation for bpower. sim() in the HMisc package, by using testBinomial() you can see that the formulas without any continuity correction are quite accurate. In fact, Type I error for a continuity-corrected test is significantly lower (Gordon and Watson, 1996) than the nominal rate. Thus, as a default no continuity corrections are performed.
varBinomial computes blinded estimates of the variance of the estimate of 1 ) event rate differences, 2) logarithm of the risk ratio, or 3) logarithm of the odds ratio. This is intended for blinded sample size re-estimation for comparative trials with a binary outcome.

Testing is 2 -sided when a Chi-square statistic is used and 1 -sided when a Z-statistic is used. Thus, these 2 options will produce substantially different results, in general. For non-inferiority, 1 -sided testing is appropriate.
You may wish to round sample sizes up using ceiling().
Farrington and Manning (1990) begin with event rates p1 and p2 under the alternative hypothesis and a difference between these rates under the null hypothesis, delta0. From these values, actual rates under the null hypothesis are computed, which are labeled p10 and p20 when outtype=3. The rates p 1 and p 2 are used to compute a variance for a Z-test comparing rates under the alternative
hypothesis, while p10 and p20 are used under the null hypothesis. This computational method is also used to estimate variances in varBinomial () based on the overall event rate observed and the input treatment difference specified in delta0.
Sample size with scale="Difference" produces an error if p1-p2=delta0. Normally, the alternative hypothesis under consideration would be p1-p2-delta0 $\gg 0 \$$. However, the alternative can have p1-p2-delta0\$<0\$.

## Usage

```
ciBinomial(x1, x2, n1, n2, alpha = 0.05, adj = 0, scale = "Difference")
nBinomial(
    p1,
    p2,
    alpha = 0.025,
    beta = 0.1,
    delta0 = 0,
    ratio = 1,
    sided = 1,
    outtype = 1,
    scale = "Difference",
    n = NULL
)
simBinomial(
    p1,
    p2,
    n1,
    n2,
    delta0 = 0,
    nsim = 10000,
    chisq = 0,
    adj = 0,
    scale = "Difference"
)
testBinomial(
    x1,
    x2,
    n1,
    n2,
    delta0 = 0,
    chisq = 0,
    adj = 0,
    scale = "Difference",
    tol = 1e-11
)
```

```
varBinomial(x, n, delta0 = 0, ratio = 1, scale = "Difference")
```


## Arguments

\(\left.\begin{array}{ll}\mathrm{x} 1 \& Number of "successes" in the control group <br>
\mathrm{x} 2 \& Number of "successes" in the experimental group <br>

\mathrm{n} 1 \& Number of observations in the control group\end{array}\right]\)| Number of observations in the experimental group |
| :--- |
| alpha |
| adj |$\quad$| type I error; see sided below to distinguish between 1- and 2-sided tests |
| :--- |
| With adj=1, the standard variance with a continuity correction is used for a |
| Miettinen and Nurminen test statistic This includes a factor of $n /(n-1)$ where |
| $n$ is the total sample size. If adj is not 1, this factor is not applied. The default is |
| adj=0 since nominal Type I error is generally conservative with adj=1 (Gordon |
| and Watson, 1996). |

A value of 0 (the default) always represents no difference between treatment groups under the null hypothesis. delta0 is interpreted differently depending on the value of the parameter scale. If scale="Difference" (the default), delta0 is the difference in event rates under the null hypothesis ( $\mathrm{p} 10-\mathrm{p} 20$ ). If scale="RR", delta0 is the logarithm of the relative risk of event rates (p10 / p20) under the null hypothesis. If scale="LNOR", delta0 is the difference in natural logarithm of the odds-ratio under the null hypothesis $\log (\mathrm{p} 10 /(1$ -p10)) $-\log (p 20 /(1-p 20))$.
ratio sample size ratio for group 2 divided by group 1
sided $\quad 2$ for 2 -sided test, 1 for 1 -sided test
outtype nBinomial only; 1 (default) returns total sample size; 2 returns a data frame with sample size for each group ( $n 1, n 2$; if $n$ is not input as NULL, power is returned in Power; 3 returns a data frame with total sample size ( n ), sample size in each group ( $\mathrm{n} 1, \mathrm{n} 2$ ), Type I error (alpha), 1 or 2 (sided, as input), Type II error (beta), power (Power), null and alternate hypothesis standard deviations (sigma0, sigma1), input event rates ( $\mathrm{p} 1, \mathrm{p} 2$ ), null hypothesis difference in treatment group means (delta0) and null hypothesis event rates (p10, p20).
n
If power is to be computed in nBinomial(), input total trial sample size in $n$; this may be a vector. This is also the sample size in varBinomial, in which case the argument must be a scalar.
nsim The number of simulations to be performed in simBinomial ()
chisq An indicator of whether or not a chi-square (as opposed to Z) statistic is to be computed. If delta $0=0$ (default), the difference in event rates divided by its standard error under the null hypothesis is used. Otherwise, a Miettinen and Nurminen chi-square statistic for a $2 \times 2$ table is used.

| tol | Default should probably be used; this is used to deal with a rounding issue in |
| :--- | :--- |
| interim calculations |  |

## Value

testBinomial() and simBinomial() each return a vector of either Chi-square or Z test statistics. These may be compared to an appropriate cutoff point (e.g., qnorm(.975) for normal or qchisq(. 95,1 ) for chi-square).
ciBinomial() returns a data frame with 1 row with a confidence interval; variable names are lower and upper.
varBinomial() returns a vector of (blinded) variance estimates of the difference of event rates (scale="Difference"), logarithm of the odds-ratio (scale="OR") or logarithm of the risk-ratio (scale="RR").
With the default outtype=1, nBinomial() returns a vector of total sample sizes is returned. With outtype $=2$, nBinomial () returns a data frame containing two vectors $n 1$ and $n 2$ containing sample sizes for groups 1 and 2, respectively; if n is input, this option also returns the power in a third vector, Power. With outtype=3, nBinomial() returns a data frame with the following columns:

| n | A vector with total samples size required for each event rate comparison specified |
| :---: | :---: |
| n1 | A vector of sample sizes for group 1 for each event rate comparison specified |
| n2 | A vector of sample sizes for group 2 for each event rate comparison specified |
| alpha | As input |
| sided | As input |
| beta | As input; if n is input, this is computed |
| Power | If $n=$ NULL on input, this is 1 -beta; otherwise, the power is computed for each sample size input |
| sigma0 | A vector containing the standard deviation of the treatment effect difference under the null hypothesis times sqrt( $n$ ) when scale="Difference" or scale="OR"; when scale="RR", this is the standard deviation time sqrt( $n$ ) for the numerator of the Farrington-Manning test statistic $\times 1-\exp ($ delta0) $\times x 2$. |
| sigma1 | A vector containing the values as sigma0, in this case estimated under the alternative hypothesis. |
| p1 | As input |
| p2 | As input |
| p10 | group 1 event rate used for null hypothesis |
| p20 | group 2 event rate used for null hypothesis |

Author(s)
Keaven Anderson <keaven\_anderson@merck.com>

## References

Farrington, CP and Manning, G (1990), 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: 1447-1454.
Fleiss, JL, Tytun, A and Ury (1980), A simple approximation for calculating sample sizes for comparing independent proportions. Biometrics;36:343-346.
Gordon, I and Watson R (1985), The myth of continuity-corrected sample size formulae. Biometrics; 52: 71-76.
Miettinen, O and Nurminen, M (1985), Comparative analysis of two rates. Statistics in Medicine; 4 : 213-226.

## See Also

Normal, uniroot

## Examples

```
# Compute z-test test statistic comparing 39/500 to 13/500
# use continuity correction in variance
x <- testBinomial(x1 = 39, x2 = 13, n1 = 500, n2 = 500, adj = 1)
x
pnorm(x, lower.tail = FALSE)
# Compute with unadjusted variance
x0 <- testBinomial(x1 = 39, x2 = 23, n1 = 500, n2 = 500)
x0
pnorm(x0, lower.tail = FALSE)
# Perform 50k simulations to test validity of the above
# asymptotic p-values
# (you may want to perform more to reduce standard error of estimate)
sum(as.double(x0) <=
    simBinomial(p1 = .078, p2 = .078, n1 = 500, n2 = 500, nsim = 10000)) / 10000
sum(as.double(x0) <=
    simBinomial(p1 = .052, p2 = .052, n1 = 500, n2 = 500, nsim = 10000)) / 10000
# Perform a non-inferiority test to see if p2=400 / 500 is within 5% of
# p1=410 / 500 use a z-statistic with unadjusted variance
x <- testBinomial(x1 = 410, x2 = 400, n1 = 500, n2 = 500, delta0 = -.05)
x
pnorm(x, lower.tail = FALSE)
# since chi-square tests equivalence (a 2-sided test) rather than
# non-inferiority (a 1-sided test),
# the result is quite different
pchisq(testBinomial(
    x1 = 410, x2 = 400, n1 = 500, n2 = 500, delta0 = -.05,
    chisq = 1, adj = 1
), 1, lower.tail = FALSE)
```

```
# now simulate the z-statistic witthout continuity corrected variance
sum(qnorm(.975) <=
    simBinomial(p1 = . 8, p2 = . 8, n1 = 500, n2 = 500, nsim = 100000)) / 100000
# compute a sample size to show non-inferiority
# with 5% margin, 90% power
nBinomial(p1 = .2, p2 = .2, delta0 = .05, alpha = .025, sided = 1, beta = .1)
# assuming a slight advantage in the experimental group lowers
# sample size requirement
nBinomial(p1 = .2, p2 = . 19, delta0 = .05, alpha = .025, sided = 1, beta = . 1)
# compute a sample size for comparing 15% vs 10% event rates
# with 1 to 2 randomization
nBinomial(p1 = . 15, p2 = .1, beta = .2, ratio = 2, alpha = .05)
# now look at total sample size using 1-1 randomization
n <- nBinomial(p1 = . 15, p2 = .1, beta = .2, alpha = .05)
n
# check if inputing sample size returns the desired power
nBinomial(p1 = . 15, p2 = .1, beta = .2, alpha = . 05, n = n)
# re-do with alternate output types
nBinomial(p1 = . 15, p2 = . 1, beta = .2, alpha = .05, outtype = 2)
nBinomial(p1 = . 15, p2 = .1, beta = .2, alpha = .05, outtype = 3)
# look at power plot under different control event rate and
# relative risk reductions
library(dplyr)
library(ggplot2)
p1 <- seq(.075, .2, .000625)
len <- length(p1)
p2 <- c(p1 * . 75, p1 * 2/3, p1 * . 6, p1 * .5)
Reduction <- c(rep("25 percent", len), rep("33 percent", len),
                    rep("40 percent", len), rep("50 percent", len))
df <- tibble(p1 = rep(p1, 4), p2, Reduction) %>%
    mutate(`Sample size` = nBinomial(p1, p2, beta = . 2, alpha = .025, sided = 1))
ggplot(df, aes(x = p1, y = `Sample size`, col = Reduction)) +
    geom_line() +
    xlab("Control group event rate") +
    ylim(0,6000) +
    ggtitle("Binomial sample size computation for 80 pct power")
# compute blinded estimate of treatment effect difference
x1 <- rbinom(n = 1, size = 100, p = .2)
x2 <- rbinom(n = 1, size = 200, p = .1)
# blinded estimate of risk difference variance
varBinomial(x = x1 + x2, n = 300, ratio = 2, delta0 = 0)
# blinded estimate of log-risk-ratio variance
varBinomial(x = x1 + x2, n = 300, ratio = 2, delta0 = 0, scale = "RR")
# blinded estimate of log-odds-ratio variance
varBinomial(x = x1 + x2, n = 300, ratio = 2, delta0 = 0, scale = "OR")
```

condPower Sample size re-estimation based on conditional power

## Description

$\operatorname{ssrCP}()$ adapts 2-stage group sequential designs to 2-stage sample size re-estimation designs based on interim analysis conditional power. This is a simple case of designs developed by Lehmacher and Wassmer, Biometrics (1999). The conditional power designs of Bauer and Kohne (1994), Proschan and Hunsberger (1995), Cui, Hung and Wang (1999) and Liu and Chi (2001), Gao, Ware and Mehta (2008), and Mehta and Pocock (2011). Either the estimated treatment effect at the interim analysis or any chosen effect size can be used for sample size re-estimation. If not done carefully, these designs can be very inefficient. It is probably a good idea to compare any design to a simpler group sequential design; see, for example, Jennison and Turnbull (2003). The a assumes a small Type I error is included with the interim analysis and that the design is an adaptation from a 2 -stage group sequential design Related functions include 3 pre-defined combination test functions (z2NC, z2Z, z2Fisher) that represent the inverse normal combination test (Lehmacher and Wassmer, 1999), the sufficient statistic for the complete data, and Fisher's combination test. Power. ssrCP computes unconditional power for a conditional power design derived by ssrCP.
condPower is a supportive routine that also is interesting in its own right; it computes conditional power of a combination test given an interim test statistic, stage 2 sample size and combination test statistic. While the returned data frame should make general plotting easy, the function plot. $\operatorname{ssrCP}()$ prints a plot of study sample size by stage 1 outcome with multiple x -axis labels for stage 1 z -value, conditional power, and stage 1 effect size relative to the effect size for which the underlying group sequential design was powered.
Sample size re-estimation using conditional power and an interim estimate of treatment effect was proposed by several authors in the 1990's (see references below). Statistical testing for these original methods was based on combination tests since Type I error could be inflated by using a sufficient statistic for testing at the end of the trial. Since 2000, more efficient variations of these conditional power designs have been developed. Fully optimized designs have also been derived (Posch et all, 2003, Lokhnygina and Tsiatis, 2008). Some of the later conditional power methods allow use of sufficient statistics for testing (Chen, DeMets and Lan, 2004, Gao, Ware and Mehta, 2008, and Mehta and Pocock, 2011).

The methods considered here are extensions of 2-stage group sequential designs that include both an efficacy and a futility bound at the planned interim analysis. A maximum fold-increase in sample size (maxinc)from the supplied group sequential design ( $x$ ) is specified by the user, as well as a range of conditional power (cpadj) where sample size should be re-estimated at the interim analysis, 1-the targeted conditional power to be used for sample size re-estimation (beta) and a combination test statistic (z2) to be used for testing at the end of the trial. The input value overrun represents incremental enrollment not included in the interim analysis that is not included in the analysis; this is used for calculating the required number of patients enrolled to complete the trial.

Whereas most of the methods proposed have been based on using the interim estimated treatment effect size (default for ssrCP), the variable theta allows the user to specify an alternative; Liu and Chi (2001) suggest that using the parameter value for which the trial was originally powered is a good choice.

## Usage

```
condPower (
        z1,
        n2,
        z2 = z2NC,
        theta \(=\) NULL,
        \(x=\) gsDesign \((k=2\), timing \(=0.5\), beta \(=\) beta \()\),
)
```

    \(\operatorname{ssrCP}(\)
        z1,
        theta = NULL,
        maxinc \(=2\),
        overrun = 0,
        beta = x\$beta,
        cpadj \(=c(0.5,1-\) beta \()\),
        \(x=\operatorname{gsDesign}(k=2\), timing \(=0.5)\),
        z2 = z2NC,
        ...
    )
    \#\# S3 method for class 'ssrCP'
    plot
        x ,
        z1ticks = NULL,
        mar = c(7, 4, 4, 4) + 0.1,
        ylab = "Adapted sample size",
        xlaboffset \(=-0.2\),
        lty = 1,
        col \(=1\),
        ...
    )
    z2NC(z1, x, ...)
    \(z 2 Z(z 1, x, n 2=x \$ n . I[2]-x \$ n . I[1], \ldots)\)
    z2Fisher(z1, x, ...)
    Power. \(\operatorname{ssrCP}(x\), theta \(=\) NULL, delta \(=\) NULL, \(r=18)\)
    
## Arguments

Scalar or vector with interim standardized Z-value(s). Input of multiple values makes it easy to plot the revised sample size as a function of the interim test statistic.
n2
stage 2 sample size to be used in computing sufficient statistic when combining

| z2 | stage 1 and 2 test statistics. <br> a combination function that returns a test statistic cutoff for the stage 2 test based <br> in the interim test statistic supplied in z1, the design $x$ and the stage 2 sample <br> size. |
| :--- | :--- |
| If NULL (default), conditional power calculation will be based on estimated in- |  |
| terim treatment effect. Otherwise, theta is the standardized effect size used for |  |
| conditional power calculation. Using the alternate hypothesis treatment effect |  |
| can be more efficient than the estimated effect size; see Liu and Chi, Biometrics |  |
| (2001). |  |

## Value

ssrCP returns a list with the following items:
$x \quad$ As input.
z2fn As input in z2.
theta standardize effect size used for conditional power; if NULL is input, this is computed as $\mathrm{z} 1 / \operatorname{sqrt}(\mathrm{n} 1)$ where n 1 is the stage 1 sample size.
maxinc As input.
overrun As input.
beta As input.
cpadj As input.
dat A data frame containing the input z1 values, computed cutoffs for the standard normal test statistic based solely on stage 2 data (z2), stage 2 sample sizes ( n 2 ), stage 2 conditional power (CP), standardize effect size used for conditional power calculation (theta), and the natural parameter value corresponding to theta (delta). The relation between theta and delta is determined by the delta0 and delta1 values from $x:$ delta $=$ delta $0+$ theta (delta1-delta0).

## Author(s)

Keaven Anderson <keaven_anderson@merck.>

## References

Bauer, Peter and Kohne, F., Evaluation of experiments with adaptive interim analyses, Biometrics, 50:1029-1041, 1994.

Chen, YHJ, DeMets, DL and Lan, KKG. Increasing the sample size when the unblinded interim result is promising, Statistics in Medicine, 23:1023-1038, 2004.
Gao, P, Ware, JH and Mehta, C, Sample size re-estimation for adaptive sequential design in clinical trials, Journal of Biopharmaceutical Statistics", 18:1184-1196, 2008.

Jennison, C and Turnbull, BW. Mid-course sample size modification in clinical trials based on the observed treatment effect. Statistics in Medicine, 22:971-993", 2003.
Lehmacher, W and Wassmer, G. Adaptive sample size calculations in group sequential trials, Biometrics, 55:1286-1290, 1999.
Liu, Q and Chi, GY., On sample size and inference for two-stage adaptive designs, Biometrics, 57:172-177, 2001.
Mehta, C and Pocock, S. Adaptive increase in sample size when interim results are promising: A practical guide with examples, Statistics in Medicine, 30:3267-3284, 2011.

## See Also

```
gsDesign
```


## Examples

```
library(ggplot2)
# quick trick for simple conditional power based on interim z-value, stage 1 and 2 sample size
# assumed treatment effect and final alpha level
# and observed treatment effect
alpha <- . 01 # set final nominal significance level
timing <- . }6\mathrm{ # set proportion of sample size, events or statistical information at IA
n2 <- 40 # set stage 2 sample size events or statistical information
hr <- . }6\mathrm{ # for this example we will derive conditional power based on hazard ratios
n.fix<- nEvents(hr=hr,alpha=alpha) # you could otherwise make n.fix an arbitrary positive value
```

```
# this just derives a group sequential design that should not change sample size from n.fix
# due to stringent IA bound
x <- gsDesign(k=2,n.fix=n.fix,alpha=alpha,test.type=1,sfu=sfHSD,
sfupar=-20, timing=timing,delta1=log(hr))
# derive effect sizes for which you wish to compute conditional power
hrpostIA = seq(.4,1,.05)
# in the following, we convert HR into standardized effect size based on the design in x
powr <- condPower(x=x,z1=1,n2=x$n.I[2]-x$n.I[1],theta=log(hrpostIA)/x$delta1*x$theta[2])
qplot(hrpostIA, condPower(x=x,z1=1,n2=x$n.I[2]-x$n.I[1],theta=log(hrpostIA)/x$delta1*x$theta[2]),
geom="line",xlab="HR post IA",ylab="Conditional power",
main ="Conditional power as a function of assumed HR")
# Following is a template for entering parameters for ssrCP
# Natural parameter value null and alternate hypothesis values
delta0 <- 0
delta1 <- 1
# timing of interim analysis for underlying group sequential design
timing <- . 5
# upper spending function
sfu <- sfHSD
# upper spending function paramater
sfupar <- -12
# maximum sample size inflation
maxinflation <- 2
# assumed enrollment overrrun at IA
overrun <- 25
# interim z-values for plotting
z <- seq(0, 4, .025)
# Type I error (1-sided)
alpha <- . }02
# Type II error for design
beta <- . }
# Fixed design sample size
n.fix <- 100
# conditional power interval where sample
# size is to be adjusted
cpadj <- c(.3, .9)
# targeted Type II error when adapting sample size
betastar <- beta
# combination test (built-in options are: z2Z, z2NC, z2Fisher)
z2 <- z2NC
# use the above parameters to generate design
# generate a 2-stage group sequential design with
x <- gsDesign(
    k = 2, n.fix = n.fix, timing = timing, sfu = sfu, sfupar = sfupar,
    alpha = alpha, beta = beta, delta0 = delta0, delta1 = delta1
)
# extend this to a conditional power design
xx <- ssrCP(
    x = x, z1 = z, overrun = overrun, beta = betastar, cpadj = cpadj,
    maxinc = maxinflation, z2 = z2
)
```

```
# plot the stage 2 sample size
plot(xx)
# demonstrate overlays on this plot
# overlay with densities for z1 under different hypotheses
lines(z, 80 + 240 * dnorm(z, mean = 0), col = 2)
lines(z, 80 + 240 * dnorm(z, mean = sqrt(x$n.I[1]) * x$theta[2]), col = 3)
lines(z, 80 + 240 * dnorm(z, mean = sqrt(x$n.I[1]) * x$theta[2] / 2), col = 4)
lines(z, 80 + 240 * dnorm(z, mean = sqrt(x$n.I[1]) * x$theta[2] * .75), col = 5)
axis(side = 4, at = 80 + 240 * seq(0, .4, .1), labels = as.character(seq(0, .4, .1)))
mtext(side = 4, expression(paste("Density for ", z[1])), line = 2)
text(x = 1.5, y = 90, col = 2, labels = expression(paste("Density for ", theta, "=0")))
text(x = 3.00, y = 180, col = 3, labels = expression(paste("Density for ", theta, "=",
    theta[1])))
text(x = 1.00, y = 180, col = 4, labels = expression(paste("Density for ", theta, "=",
    theta[1], "/2")))
text(x = 2.5, y = 140, col = 5, labels = expression(paste("Density for ", theta, "=",
    theta[1], "*.75")))
# overall line for max sample size
nalt <- xx$maxinc * x$n.I[2]
lines(x = par("usr")[1:2], y = c(nalt, nalt), lty = 2)
# compare above design with different combination tests
# use sufficient statistic for final testing
xxZ <- ssrCP(
    x = x, z1 = z, overrun = overrun, beta = betastar, cpadj = cpadj,
    maxinc = maxinflation, z2 = z2Z
)
# use Fisher combination test for final testing
xxFisher <- ssrCP(
    x = x, z1 = z, overrun = overrun, beta = betastar, cpadj = cpadj,
    maxinc = maxinflation, z2 = z2Fisher
)
# combine data frames from these designs
y <- rbind(
    data.frame(cbind(xx$dat, Test = "Normal combination")),
    data.frame(cbind(xxZ$dat, Test = "Sufficient statistic")),
    data.frame(cbind(xxFisher$dat, Test = "Fisher combination"))
)
# plot stage 2 statistic required for positive combination test
ggplot2::ggplot(data = y, ggplot2::aes(x = z1, y = z2, col = Test)) +
ggplot2::geom_line()
# plot total sample size versus stage 1 test statistic
ggplot2::ggplot(data = y, ggplot2::aes(x = z1, y = n2, col = Test)) +
ggplot2::geom_line()
# check achieved Type I error for sufficient statistic design
Power.ssrCP(x = xxZ, theta = 0)
# compare designs using observed vs planned theta for conditional power
xxtheta1 <- ssrCP(
    x = x, z1 = z, overrun = overrun, beta = betastar, cpadj = cpadj,
    maxinc = maxinflation, z2 = z2, theta = x$delta
)
# combine data frames for the 2 designs
```

```
y <- rbind(
    data.frame(cbind(xx$dat, "CP effect size" = "Obs. at IA")),
    data.frame(cbind(xxtheta1$dat, "CP effect size" = "Alt. hypothesis"))
)
# plot stage 2 sample size by design
ggplot2::ggplot(data = y, ggplot2::aes(x = z1, y = n2, col = CP.effect.size)) +
ggplot2::geom_line()
# compare power and expected sample size
y1 <- Power.ssrCP(x = xx)
y2 <- Power.ssrCP(x = xxtheta1)
# combine data frames for the 2 designs
y3 <- rbind(
    data.frame(cbind(y1, "CP effect size" = "Obs. at IA")),
    data.frame(cbind(y2, "CP effect size" = "Alt. hypothesis"))
)
# plot expected sample size by design and effect size
ggplot2::ggplot(data = y3, ggplot2::aes(x = delta, y = en, col = CP.effect.size)) +
ggplot2::geom_line() +
ggplot2::xlab(expression(delta)) +
ggplot2::ylab("Expected sample size")
# plot power by design and effect size
ggplot2::ggplot(data = y3, ggplot2::aes(x = delta, y = Power, col = CP.effect.size)) +
ggplot2::geom_line() +
ggplot2::xlab(expression(delta))
```


## Description

eEvents() is used to calculate the expected number of events for a population with a time-to-event endpoint. It is based on calculations demonstrated in Lachin and Foulkes (1986) and is fundamental in computations for the sample size method they propose. Piecewise exponential survival and dropout rates are supported as well as piecewise uniform enrollment. A stratified population is allowed. Output is the expected number of events observed given a trial duration and the above rate parameters.
eEvents() produces an object of class eEvents with the number of subjects and events for a set of pre-specified trial parameters, such as accrual duration and follow-up period. The underlying power calculation is based on Lachin and Foulkes (1986) method for proportional hazards assuming a fixed underlying hazard ratio between 2 treatment groups. The method has been extended here to enable designs to test non-inferiority. Piecewise constant enrollment and failure rates are assumed and a stratified population is allowed. See also nSurvival for other Lachin and Foulkes (1986) methods assuming a constant hazard difference or exponential enrollment rate.
print.eEvents() formats the output for an object of class eEvents and returns the input value.

## Usage

eEvents(

```
    lambda = 1,
    eta = 0,
    gamma = 1,
    R = 1,
    S = NULL,
    T = 2,
    Tfinal = NULL,
    minfup = 0,
    digits = 4
)
## S3 method for class 'eEvents'
print(x, digits = 4, ...)
```


## Arguments

lambda scalar, vector or matrix of event hazard rates; rows represent time periods while columns represent strata; a vector implies a single stratum.
eta scalar, vector or matrix of dropout hazard rates; rows represent time periods while columns represent strata; if entered as a scalar, rate is constant across strata and time periods; if entered as a vector, rates are constant across strata.
gamma a scalar, vector or matrix of rates of entry by time period (rows) and strata (columns); if entered as a scalar, rate is constant across strata and time periods; if entered as a vector, rates are constant across strata.
$R \quad a \quad$ scalar or vector of durations of time periods for recruitment rates specified in rows of gamma. Length is the same as number of rows in gamma. Note that the final enrollment period is extended as long as needed.

S
a scalar or vector of durations of piecewise constant event rates specified in rows of lambda, eta and etaE; this is NULL if there is a single event rate per stratum (exponential failure) or length of the number of rows in lambda minus 1 , otherwise.

T time of analysis; if Tfinal=NULL, this is also the study duration.
Tfinal Study duration; if NULL, this will be replaced with $T$ on output.
minfup time from end of planned enrollment (sum(R) from output value of $R$ ) until Tfinal.
digits which controls number of digits for printing.
$x \quad$ an object of class eEvents returned from eEvents ().
$\ldots \quad$ Other arguments that may be passed to the generic print function.
Value
eEvents() and print.eEvents() return an object of class eEvents which contains the following items:
lambda as input; converted to a matrix on output.
eta as input; converted to a matrix on output.

| gamma | as input. |
| :--- | :--- |
| R | as input. |
| S | as input. |
| T | as input. |
| Tfinal | planned duration of study. |
| minfup | as input. |
| d | expected number of events. |
| n | expected sample size. |
| digits | as input. |

## Author(s)

Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)

## References

Lachin JM and Foulkes MA (1986), 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, 507-519.

Bernstein D and Lagakos S (1978), Sample size and power determination for stratified clinical trials. Journal of Statistical Computation and Simulation, 8:65-73.

## See Also

gsDesign package overview, plot.gsDesign, gsDesign, gsHR, nSurvival

## Examples

```
# 3 enrollment periods, 3 piecewise exponential failure rates
str(eEvents(
    lambda = c(.05, .02, .01), eta = .01, gamma = c(5, 10, 20),
    R = c(2, 1, 2), S = c(1, 1), T = 20
))
# control group for example from Berstein and Lagakos (1978)
lamC <- c(1, .8, .5)
n <- eEvents(
    lambda = matrix(c(lamC, lamC * 2 / 3), ncol = 6), eta = 0,
    gamma = matrix(.5, ncol = 6), R = 2, T = 4
)
```


## Description

gsBinomialExact computes power/Type I error and expected sample size for a group sequential design in a single-arm trial with a binary outcome. This can also be used to compare event rates in two-arm studies. The print function has been extended using print.gsBinomialExact to print gsBinomialExact objects. Similarly, a plot function has been extended using plot.gsBinomialExact to plot gsBinomialExact objects.
binomialSPRT computes a truncated binomial sequential probability ratio test (SPRT) which is a specific instance of an exact binomial group sequential design for a single arm trial with a binary outcome.
based on predictive probability.
nBinomial1Sample uses exact binomial calculations to compute power and sample size for single arm binomial experiments.
gsBinomialExact is based on the book "Group Sequential Methods with Applications to Clinical Trials," Christopher Jennison and Bruce W. Turnbull, Chapter 12, Section 12.1.2 Exact Calculations for Binary Data. This computation is often used as an approximation for the distribution of the number of events in one treatment group out of all events when the probability of an event is small and sample size is large.
An object of class gsBinomialExact is returned. On output, the values of theta input to gsBinomialExact will be the parameter values for which the boundary crossing probabilities and expected sample sizes are computed.

Note that a[1] equal to -1 lower bound at n.I[1] means 0 successes continues at interim $1 ; \mathrm{a}[2]==0$ at interim 2 means 0 successes stops trial for futility at 2 nd analysis. For final analysis, set $\mathrm{a}[\mathrm{k}]$ equal to $\mathrm{b}[\mathrm{k}]-1$ to incorporate all possibilities into non-positive trial; see example.
The sequential probability ratio test (SPRT) is a sequential testing scheme allowing testing after each observation. This likelihood ratio is used to determine upper and lower cutoffs which are linear and parallel in the number of responses as a function of sample size. binomialSPRT produces a variation the the SPRT that tests only within a range of sample sizes. While the linear SPRT bounds are continuous, actual bounds are the integer number of response at or beyond each linear bound for each sample size where testing is performed. Because of the truncation and discretization of the bounds, power and Type I error achieve will be lower than the nominal levels specified by alpha and beta which can be altered to produce desired values that are achieved by the planned sample size. See also example that shows computation of Type I error when futility bound is considered non-binding.
Note that if the objective of a design is to demonstrate that a rate (e.g., failure rate) is lower than a certain level, two approaches can be taken. First, 1 minus the failure rate is the success rate and this can be used for planning. Second, the role of beta becomes to express Type I error and alpha is used to express Type II error.

Plots produced include boundary plots, expected sample size, response rate at the boundary and power.
gsBinomial1Sample uses exact binomial computations based on the base R functions qbinom() and pbinom(). The tabular output may be convenient for plotting. Note that input variables are largely not checked, so the user is largely responsible for results; it is a good idea to do a run with outtype $=3$ to check that you have done things appropriately. If $n$ is not ordered (a bad idea) or not sequential (maybe OK), be aware of possible consequences.
nBinomial1Sample is based on code from Marc Schwartz marc_schwartz@me.com. The possible sample size vector $n$ needs to be selected in such a fashion that it covers the possible range of values that include the true minimum. NOTE: the one-sided evaluation of significance is more conservative than using the 2 -sided exact test in binom. test.

## Usage

```
gsBinomialExact(
        k = 2,
        theta = c(0.1, 0.2),
        n.I = c(50, 100),
        a = c(3, 7),
        b = c(20, 30)
    )
    binomialSPRT(
        p0 = 0.05,
        p1 = 0.25,
        alpha = 0.1,
        beta = 0.15,
        minn = 10,
        maxn = 35
    )
    ## S3 method for class 'gsBinomialExact'
    plot(x, plottype = 1, ...)
    ## S3 method for class 'binomialSPRT'
    plot(x, plottype = 1, ...)
    nBinomial1Sample(
        p0 = 0.9,
        p1 = 0.95,
        alpha = 0.025,
        beta = NULL,
        n = 200:250,
        outtype = 1,
        conservative = FALSE
)
```


## Arguments

k
Number of analyses planned, including interim and final.

| theta | Vector of possible underling binomial probabilities for a single binomial sample. |
| :---: | :---: |
| n. I | Sample size at analyses (increasing positive integers); vector of length k. |
| a | Number of "successes" required to cross lower bound cutoffs to reject p1 in favor of p 0 at each analysis; vector of length $\mathrm{k} ;-1$ means no lower bound. |
| b | Number of "successes" required to cross upper bound cutoffs for rejecting p0 in favor of $p 1$ at each analysis; vector of length $k$. |
| p0 | Lower of the two response (event) rates hypothesized. |
| p1 | Higher of the two response (event) rates hypothesized. |
| alpha | Nominal probability of rejecting response (event) rate p0 when it is true. |
| beta | Nominal probability of rejecting response (event) rate p 1 when it is true. If NULL, Type II error will be computed for all input values of n and output will be in a data frame. |
| minn | Minimum sample size at which sequential testing begins. |
| maxn | Maximum sample size. |
| x | Item of class gsBinomialExact or binomialSPRT for print.gsBinomialExact. Item of class gsBinomialExact for plot.gsBinomialExact. Item of class binomialSPRT for item of class plot.binomialSPRT. |
| plottype | 1 produces a plot with counts of response at bounds (for binomialSPRT, also produces linear SPRT bounds); 2 produces a plot with power to reject null and alternate response rates as well as the probability of not crossing a bound by the maximum sample size; 3 produces a plot with the response rate at the boundary as a function of sample size when the boundary is crossed; 6 produces a plot of the expected sample size by the underlying event rate (this assumes there is no enrollment beyond the sample size where the boundary is crossed). |
|  | arguments passed through to ggplot. |
| n | sample sizes to be considered for nBinomial1Sample. These should be ordered from smallest to largest and be $>0$. |
| outtype | Operative when beta ! = NULL. 1 means routine will return a single integer sample size while for output=2a data frame is returned (see Value); note that this not operative is beta is NULL in which case a data table is returned with Type II error and power for each input value of $n$. |
| conservative | operative when outtype=1 or 2 and beta ! = NULL. Default FALSE selects minimum sample size for which power is at least 1-beta. When conservative=TRUE, the minimum sample sample size for which power is at least 1-beta and there is no larger sample size in the input n where power is less than 1 -beta. |

## Value

gsBinomialExact() returns a list of class gsBinomialExact and gsProbability (see example); when displaying one of these objects, the default function to print is print.gsProbability(). The object returned from gsBinomialExact () contains the following elements:

| $k$ | As input. |
| :--- | :--- |
| theta | As input. |

n.I | As input. |
| :--- |
| A list containing two elements: bound is as input in a and prob is a matrix of |
| boundary crossing probabilities. Element i , j contains the boundary crossing |
| probability at analysis i for the j-th element of theta input. All boundary |
| crossing is assumed to be binding for this computation; that is, the trial must |
| stop if a boundary is crossed. |
| A list of the same form as lower containing the upper bound and upper boundary |
| crossing probabilities. |

| A vector of the same length as theta containing expected sample sizes for the |
| :--- |
| upper |

trial design corresponding to each value in the vector theta.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Jon Hartzel, Yevgen Tymofyeyev and Keaven Anderson <keaven \_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Code for nBinomial1Sample was based on code developed by marc_schwartz@me.com.

## See Also

gsProbability

## Examples

```
library(ggplot2)
zz <- gsBinomialExact(
    k = 3, theta = seq (0, 1, 0.1), n.I = c(12, 24, 36),
    a = c(-1, 0, 11), b = c(5, 9, 12)
)
# let's see what class this is
class(zz)
# because of "gsProbability" class above, following is equivalent to
# \code{print.gsProbability(zz)}
zz
# also plot (see also plots below for \code{binomialSPRT})
# add lines using geom_line()
plot(zz) +
ggplot2::geom_line()
# now for SPRT examples
x <- binomialSPRT(p0 = .05, p1 = . 25, alpha = .1, beta = .2)
# boundary plot
plot(x)
# power plot
plot(x, plottype = 2)
# Response (event) rate at boundary
plot(x, plottype = 3)
# Expected sample size at boundary crossing or end of trial
plot(x, plottype = 6)
# sample size for single arm exact binomial
# plot of table of power by sample size
# note that outtype need no be specified if beta is NULL
nb1 <- nBinomial1Sample(p0 = 0.05, p1=0.2,alpha = 0.025, beta=NULL, n = 25:40)
nb1
```

```
library(scales)
ggplot2::ggplot(nb1, ggplot2::aes(x = n, y = Power)) +
ggplot2::geom_line() +
ggplot2::geom_point() +
ggplot2::scale_y_continuous(labels = percent)
# simple call with same parameters to get minimum sample size yielding desired power
nBinomial1Sample(p0 = 0.05, p1 = 0.2, alpha = 0.025, beta = .2, n = 25:40)
# change to 'conservative' if you want all larger sample
# sizes to also provide adequate power
nBinomial1Sample(p0 = 0.05, p1 = 0.2, alpha = 0.025, beta = . 2, n = 25:40, conservative = TRUE)
# print out more information for the selected derived sample size
nBinomial1Sample(p0 = 0.05, p1 = 0.2, alpha = 0.025, beta = . 2, n = 25:40, conservative = TRUE,
    outtype = 2)
# what happens if input sample sizes not sufficient?
## Not run:
    nBinomial1Sample(p0 = 0.05, p1 = 0.2, alpha = 0.025, beta = . 2, n = 25:30)
## End(Not run)
```

gsBound Boundary derivation-low level

## Description

gsBound() and gsBound1 () are lower-level functions used to find boundaries for a group sequential design. They are not recommended (especially gsBound1()) for casual users. These functions do not adjust sample size as gsDesign() does to ensure appropriate power for a design.
gsBound() computes upper and lower bounds given boundary crossing probabilities assuming a mean of 0 , the usual null hypothesis. gsBound1 () computes the upper bound given a lower boundary, upper boundary crossing probabilities and an arbitrary mean (theta).
The function gsBound1 () requires special attention to detail and knowledge of behavior when a design corresponding to the input parameters does not exist.

## Usage

gsBound(I, trueneg, falsepos, tol $=1 \mathrm{e}-06, r=18$, printerr $=0$ )
gsBound1(theta, $I, a$, probhi, tol $=1 \mathrm{e}-06, r=18$, printerr $=0$ )

## Arguments

I
trueneg
falsepos

Vector containing statistical information planned at each analysis.
Vector of desired probabilities for crossing upper bound assuming mean of 0 .
Vector of desired probabilities for crossing lower bound assuming mean of 0 .

| tol | Tolerance for error (scalar; default is 0.000001$)$. Normally this will not be <br> changed by the user. This does not translate directly to number of digits of <br> accuracy, so use extra decimal places. |
| :--- | :--- |
| $r$ | Single integer value controlling grid for numerical integration as in Jennison <br> and Turnbull (2000); default is 18, range is 1 to 80 . Larger values provide larger <br> number of grid points and greater accuracy. Normally $r$ will not be changed by <br> the user. |
| printerr | If this scalar argument set to 1, this will print messages from underlying C pro- <br> gram. Mainly intended to notify user when an output solution does not match <br> input specifications. This is not intended to stop execution as this often occurs <br> when deriving a design in gsDesign that uses beta-spending. |
| theta | Scalar containing mean (drift) per unit of statistical information. <br> a$\quad$Vector containing lower bound that is fixed for use in gsBound1. |
| probhi | Vector of desired probabilities for crossing upper bound assuming mean of theta. |

## Value

Both routines return a list. Common items returned by the two routines are:
$k \quad$ The length of vectors input; a scalar.
theta As input in gsBound1(); 0 for gsBound().
I
As input.
a For gsbound1, this is as input. For gsbound this is the derived lower boundary required to yield the input boundary crossing probabilities under the null hypothesis.
b The derived upper boundary required to yield the input boundary crossing probabilities under the null hypothesis.
tol As input.
$r$ As input.
error Error code. 0 if no error; greater than 0 otherwise.
gsBound() also returns the following items:
rates a list containing two items:
falsepos vector of upper boundary crossing probabilities as input.
trueneg vector of lower boundary crossing probabilities as input.
gsBound1 () also returns the following items:
problo vector of lower boundary crossing probabilities; computed using input lower bound and derived upper bound.
probhi vector of upper boundary crossing probabilities as input.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

gsDesign package overview, gsDesign, gsProbability

## Examples

```
# set boundaries so that probability is . 01 of first crossing
# each upper boundary and . 02 of crossing each lower boundary
# under the null hypothesis
x <- gsBound(
    I = c(1, 2, 3) / 3, trueneg = rep(.02, 3),
    falsepos = rep(.01, 3)
)
x
# use gsBound1 to set up boundary for a 1-sided test
x <- gsBound1(
    theta = 0, I = c(1, 2, 3) / 3, a = rep (-20, 3),
    probhi = c(.001, .009, .015)
)
x$b
# check boundary crossing probabilities with gsProbability
y <- gsProbability(k = 3, theta = 0, n.I = x$I, a = x$a, b = x$b)$upper$prob
    Note that gsBound1 only computes upper bound
To get a lower bound under a parameter value theta:
            use minus the upper bound as a lower bound
            replace theta with -theta
            set probhi as desired lower boundary crossing probabilities
# Here we let set lower boundary crossing at 0.05 at each analysis
# assuming theta=2.2
y <- gsBound1(
    theta = -2.2, I = c(1, 2, 3) / 3, a = -x$b,
    probhi = rep(.05, 3)
)
y$b
# Now use gsProbability to look at design
Note that lower boundary crossing probabilities are as
specified for theta=2.2, but for theta=0 the upper boundary
crossing probabilities are smaller than originally specified
# above after first interim analysis
```

```
gsProbability(k = length(x$b), theta = c(0, 2.2), n.I = x$I, b = x$b, a = -y$b)
```

gsBoundCP Conditional Power at Interim Boundaries

## Description

gsBoundCP() computes the total probability of crossing future upper bounds given an interim test statistic at an interim bound. For each interim boundary, assumes an interim test statistic at the boundary and computes the probability of crossing any of the later upper boundaries.
See Conditional power section of manual for further clarification. See also Muller and Schaffer (2001) for background theory.

## Usage

gsBoundCP(x, theta $=$ "thetahat", $r=18)$

## Arguments

X
theta
$r$

An object of type gsDesign or gsProbability
if "thetahat" and class(x)!="gsDesign", conditional power computations for each boundary value are computed using estimated treatment effect assuming a test statistic at that boundary (zi/sqrt(x\$n.I[i]) at analysis i, interim test statistic zi and interim sample size/statistical information of $x \$ n$.I[i]). Otherwise, conditional power is computed assuming the input scalar value theta.

Integer value controlling grid for numerical integration as in Jennison and Turnbull (2000); default is 18 , range is 1 to 80 . Larger values provide larger number of grid points and greater accuracy. Normally $r$ will not be changed by the user.

## Value

A list containing two vectors, CPlo and CPhi.
CPlo A vector of length $\times \$ \mathrm{k}-1$ with conditional powers of crossing upper bounds given interim test statistics at each lower bound
CPhi A vector of length $\times \$ \mathrm{k}-1$ with conditional powers of crossing upper bounds given interim test statistics at each upper bound.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven \_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Muller, Hans-Helge and Schaffer, Helmut (2001), Adaptive group sequential designs for clinical trials: combining the advantages of adaptive and classical group sequential approaches. Biomet-rics;57:886-891.

See Also<br>gsDesign, gsProbability, gsCP

## Examples

```
# set up a group sequential design
x <- gsDesign(k = 5)
x
# compute conditional power based on interim treatment effects
gsBoundCP(x)
# compute conditional power based on original x$delta
gsBoundCP(x, theta = x$delta)
```

gsCP

Conditional and Predictive Power, Overall and Conditional Probability of Success

## Description

$\operatorname{gsCP}()$ computes conditional boundary crossing probabilities at future planned analyses for a given group sequential design assuming an interim z-statistic at a specified interim analysis. While $\operatorname{gsCP}()$ is designed toward computing conditional power for a variety of underlying parameter values, condPower is built to compute conditional power for a variety of interim test statistic values which is useful for sample size adaptation (see $s s r C P$ ). gsPP() averages conditional power across a posterior distribution to compute predictive power. gsPI() computes Bayesian prediction intervals for future analyses corresponding to results produced by $\operatorname{gsPP}()$. gsPosterior() computes the posterior density for the group sequential design parameter of interest given a prior density and an interim outcome that is exact or in an interval. gsPOS() computes the probability of success for a trial using a prior distribution to average power over a set of theta values of interest. gsCPOS() assumes no boundary has been crossed before and including an interim analysis of interest, and computes the probability of success based on this event. Note that $\operatorname{gsCP}()$ and $\operatorname{gsPP}()$ take only the interim test statistic into account in computing conditional probabilities, while gsCPOS() conditions on not crossing any bound through a specified interim analysis.
See Conditional power section of manual for further clarification. See also Muller and Schaffer (2001) for background theory.

For $\operatorname{gsPP}(), \operatorname{gsPI}(), \operatorname{gsPOS}()$ and $g s C P O S()$, the prior distribution for the standardized parameter theta () for a group sequential design specified through a gsDesign object is specified through the arguments theta and wgts. This can be a discrete or a continuous probability density function. For a discrete function, generally all weights would be 1 . For a continuous density, the wgts would contain integration grid weights, such as those provided by normalGrid.
For gsPosterior, a prior distribution in prior must be composed of the vectors $z$ density. The vector $z$ contains points where the prior is evaluated and density the corresponding density or, for a discrete distribution, the probabilities of each point in $z$. Densities may be supplied as from normalGrid() where grid weights for numerical integration are supplied in gridwgts. If gridwgts are not supplied, they are defaulted to 1 (equal weighting). To ensure a proper prior distribution, you must have sum(gridwgts * density) equal to 1 ; this is NOT checked, however.

## Usage

```
gsCP(x, theta = NULL, i = 1, zi = 0, r = 18)
gsPP(
    x,
    i = 1,
    zi = 0,
    theta = c(0, 3),
    wgts =c(0.5, 0.5),
    r = 18,
    total = TRUE
)
gsPI(
    X,
    i = 1,
    zi = 0,
    j = 2,
    level = 0.95,
    theta = c(0, 3),
    wgts = c(0.5, 0.5)
)
gsPosterior(x = gsDesign(), i = 1, zi = NULL, prior = normalGrid(), r = 18)
gsPOS(x, theta, wgts)
gsCPOS(i, x, theta, wgts)
```


## Arguments

| $x$ | An object of type gsDesign or gsProbability |
| :--- | :--- |
| theta | a vector with $\theta$ value(s) at which conditional power is to be computed; for |
| $\operatorname{gsCP}()$ if NULL, an estimated value of $\theta$ based on the interim test statistic $(\mathrm{zi} / \operatorname{sqrt}(\mathrm{x} \$ \mathrm{n} . \mathrm{I}[\mathrm{i}]))$ |  |
| as well as at $\times \$$ theta is computed. For gsPosterior, this may be a scalar or an |  |
| interval; for gsPP and gsCP, this must be a scalar. |  |


| i | analysis at which interim z-value is given; must be from 1 to $\mathrm{x} \$ \mathrm{k}-1$ |
| :---: | :---: |
| zi | interim z-value at analysis i (scalar) |
| $r$ | Integer value controlling grid for numerical integration as in Jennison and Turnbull (2000); default is 18 , range is 1 to 80 . Larger values provide larger number of grid points and greater accuracy. Normally $r$ will not be changed by the user. |
| wgts | Weights to be used with grid points in theta. Length can be one if weights are equal, otherwise should be the same length as theta. Values should be positive, but do not need to sum to 1 . |
| total | The default of total=TRUE produces the combined probability for all planned analyses after the interim analysis specified in i. Otherwise, information on each analysis is provided separately. |
| j | specific analysis for which prediction is being made; must be $>\mathrm{i}$ and no more than $\mathrm{x} \$ \mathrm{k}$ |
| level | The level to be used for Bayes credible intervals (which approach confidence intervals for vague priors). The default level=. 95 corresponds to a $95 \%$ credible interval. level=0 provides a point estimate rather than an interval. |
| prior | provides a prior distribution in the form produced by normalGrid |

## Value

$\operatorname{gsCP}()$ returns an object of the class gsProbability. Based on the input design and the interim test statistic, the output gsDesign object has bounds for test statistics computed based on solely on observations after interim i. Boundary crossing probabilities are computed for the input $\theta$ values. See manual and examples.
$\operatorname{gsPP}()$ if total==TRUE, returns a real value indicating the predictive power of the trial conditional on the interim test statistic zi at analysis $i$; otherwise returns vector with predictive power for each future planned analysis.
gsPI () returns an interval (or point estimate if level=0) indicating 100level\% credible interval for the $z$-statistic at analysis $j$ conditional on the $z$-statistic at analysis $i<j$. The interval does not consider intervending interim analyses. The probability estimate is based on the predictive distribution used for $\operatorname{gsPP}()$ and requires a prior distribution for the group sequential parameter theta specified in theta and wgts.
gsPosterior() returns a posterior distribution containing the the vector z input in prior $\$ \mathrm{z}$, the posterior density in density, grid weights for integrating the posterior density as input in prior\$gridwgts or defaulted to a vector of ones, and the product of the output values in density and gridwgts in wgts.
gsPOS() returns a real value indicating the probability of a positive study weighted by the prior distribution input for theta.
gsCPOS() returns a real value indicating the probability of a positive study weighted by the posterior distribution derived from the interim test statistic and the prior distribution input for theta conditional on an interim test statistic.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Proschan, Michael A., Lan, KK Gordon and Wittes, Janet Turk (2006), Statistical Monitoring of Clinical Trials. NY: Springer.
Muller, Hans-Helge and Schaffer, Helmut (2001), Adaptive group sequential designs for clinical trials: combining the advantages of adaptive and classical group sequential approaches. Biomet-rics;57:886-891.

## See Also

normalGrid, gsDesign, gsProbability, gsBoundCP, ssrCP, condPower

## Examples

```
library(ggplot2)
# set up a group sequential design
x <- gsDesign(k = 5)
x
# set up a prior distribution for the treatment effect
# that is normal with mean .75*x$delta and standard deviation x$delta/2
mu0 <- . }75\mathrm{ * x$delta
sigma0 <- x$delta / 2
prior <- normalGrid(mu = mu0, sigma = sigma0)
# compute POS for the design given the above prior distribution for theta
gsPOS(x = x, theta = prior$z, wgts = prior$wgts)
# assume POS should only count cases in prior where theta >= x$delta/2
gsPOS(x = x, theta = prior$z, wgts = prior$wgts * (prior$z >= x$delta / 2))
# assuming a z-value at lower bound at analysis 2, what are conditional
# boundary crossing probabilities for future analyses
# assuming theta values from x as well as a value based on the interim
# observed z
CP <- gsCP(x, i = 2, zi = x$lower$bound[2])
CP
# summing values for crossing future upper bounds gives overall
# conditional power for each theta value
CP$theta
t(CP$upper$prob) %*% c(1, 1, 1)
# compute predictive probability based on above assumptions
gsPP(x, i = 2, zi = x$lower$bound[2], theta = prior$z, wgts = prior$wgts)
```

```
# if it is known that boundary not crossed at interim 2, use
# gsCPOS to compute conditional POS based on this
gsCPOS(x = x, i = 2, theta = prior$z, wgts = prior$wgts)
# 2-stage example to compare results to direct computation
x <- gsDesign(k = 2)
z1 <- 0.5
n1 <- x$n.I[1]
n2 <- x$n.I[2] - x$n.I[1]
thetahat <- z1 / sqrt(n1)
theta <- c(thetahat, 0, x$delta)
# conditional power direct computation - comparison w gsCP
pnorm((n2 * theta + z1 * sqrt(n1) - x$upper$bound[2] * sqrt(n1 + n2)) / sqrt(n2))
gsCP(x = x, zi = z1, i = 1)$upper$prob
# predictive power direct computation - comparison w gsPP
# use same prior as above
mu0 <- . }75\mathrm{ * x$delta * sqrt(x$n.I[2])
sigma2 <- (.5 * x$delta)^2 * x$n.I[2]
prior <- normalGrid(mu = . }75\mathrm{ * x$delta, sigma = x$delta / 2)
gsPP(x = x, zi = z1, i = 1, theta = prior$z, wgts = prior$wgts)
t <- . }
z1 <- .5
b <- z1 * sqrt(t)
# direct from Proschan, Lan and Wittes eqn 3.10
# adjusted drift at n.I[2]
pnorm(((b - x$upper$bound[2]) * (1 + t * sigma2) +
    (1 - t) * (mu0 + b * sigma2)) /
    sqrt((1 - t) * (1 + sigma2) * (1 + t * sigma2)))
# plot prior then posterior distribution for unblinded analysis with i=1, zi=1
xp <- gsPosterior(x = x, i = 1, zi = 1, prior = prior)
plot(x = xp$z, y = xp$density, type = "l", col = 2, xlab = expression(theta), ylab = "Density")
points(x = x$z, y = x$density, col = 1)
# add posterior plot assuming only knowlede that interim bound has
# not been crossed at interim 1
xpb <- gsPosterior(x = x, i = 1, zi = 1, prior = prior)
lines(x = xpb$z, y = xpb$density, col = 4)
# prediction interval based in interim 1 results
# start with point estimate, followed by 90% prediction interval
gsPI(x = x, i = 1, zi = z1, j = 2, theta = prior$z, wgts = prior$wgts, level = 0)
gsPI(x = x, i = 1, zi = z1, j = 2, theta = prior$z, wgts = prior$wgts, level = .9)
```


## Description

Given an interim analysis i of a group sequential design and a vector of real values zi, gsDensity() computes an interim density function at analysis $i$ at the values in zi. For each value in zi, this interim density is the derivative of the probability that the group sequential trial does not cross a boundary prior to the i-th analysis and at the i-th analysis the interim Z-statistic is less than that value. When integrated over the real line, this density computes the probability of not crossing a bound at a previous analysis. It corresponds to the subdistribution function at analysis i that excludes the probability of crossing a bound at an earlier analysis.
The initial purpose of this routine was as a component needed to compute the predictive power for a trial given an interim result; see gsPP.
See Jennison and Turnbull (2000) for details on how these computations are performed.

## Usage

gsDensity (x, theta $=0, i=1, z i=0, r=18)$

## Arguments

| x | An object of type gsDesign or gsProbability |
| :--- | :--- |
| theta | a vector with $\theta$ value(s) at which the interim density function is to be computed. |
| i | analysis at which interim z-values are given; must be from 1 to $\mathrm{x} \$ \mathrm{k}$ |
| zi | interim z-value at analysis i (scalar) <br> r |
| Integer value controlling grid for numerical integration as in Jennison and Turn- <br> bull (2000); default is 18, range is 1 to 80. Larger values provide larger number <br> of grid points and greater accuracy. Normally $r$ will not be changed by the user. |  |

## Value

zi The input vector zi.
theta The input vector theta.
density A matrix with length(zi) rows and length(theta) columns. The subdensity function for $z[j]$, theta[m] at analysis $i$ is returned in density $[j, m]$.

Note
The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

gsDesign, gsProbability, gsBoundCP

## Examples

```
library(ggplot2)
# set up a group sequential design
x <- gsDesign()
# set theta values where density is to be evaluated
theta <- x$theta[2] * c(0, .5, 1, 1.5)
# set zi values from -1 to 7 where density is to be evaluated
zi <- seq(-3, 7, .05)
# compute subdensity values at analysis 2
y <- gsDensity(x, theta = theta, i = 2, zi = zi)
# plot sub-density function for each theta value
plot(y$zi, y$density[, 3],
    type = "l", xlab = "Z",
    ylab = "Interim 2 density", lty = 3, lwd = 2
)
lines(y$zi, y$density[, 2], lty = 2, lwd = 2)
lines(y$zi, y$density[, 1], lwd = 2)
lines(y$zi, y$density[, 4], lty = 4, lwd = 2)
title("Sub-density functions at interim analysis 2")
legend(
    x = c(3.85, 7.2), y = c(.27, .385), lty = 1:5, lwd = 2, cex = 1.5,
    legend = c(
            expression(paste(theta, "=0.0")),
            expression(paste(theta, "=0.5", delta)),
            expression(paste(theta, "=1.0", delta)),
            expression(paste(theta, "=1.5", delta))
    )
)
# add vertical lines with lower and upper bounds at analysis 2
# to demonstrate how likely it is to continue, stop for futility
# or stop for efficacy at analysis 2 by treatment effect
lines(rep(x$upper$bound[2], 2), c(0, .4), col = 2)
lines(rep(x$lower$bound[2], 2), c(0, .4), lty = 2, col = 2)
# Replicate part of figures 8.1 and 8.2 of Jennison and Turnbull text book
# O'Brien-Fleming design with four analyses
x <- gsDesign(k = 4, test.type = 2, sfu = "OF", alpha = .1, beta = .2)
z<- seq(-4.2, 4.2, .05)
d <- gsDensity(x = x, theta = x$theta, i = 4, zi = z)
plot(z, d$density[, 1], type = "l", lwd = 2, ylab = expression(paste(p[4], "(z,", theta, ")")))
```

```
lines(z, d$density[, 2], lty = 2, lwd = 2)
u <- x$upper$bound[4]
text(expression(paste(theta, "=", delta)), x = 2.2, y = .2, cex = 1.5)
text(expression(paste(theta, "=0")), x = .55, y = .4, cex = 1.5)
```

gsDesign Design Derivation

## Description

gsDesign() is used to find boundaries and trial size required for a group sequential design.
Many parameters normally take on default values and thus do not require explicit specification. One- and two-sided designs are supported. Two-sided designs may be symmetric or asymmetric. Wang-Tsiatis designs, including O'Brien-Fleming and Pocock designs can be generated. Designs with common spending functions as well as other built-in and user-specified functions for Type I error and futility are supported. Type I error computations for asymmetric designs may assume binding or non-binding lower bounds. The print function has been extended using print.gsDesign() to print gsDesign objects; see examples.

The user may ignore the structure of the value returned by gsDesign() if the standard printing and plotting suffice; see examples.
delta and n.fix are used together to determine what sample size output options the user seeks. The default, delta=0 and n.fix=1, results in a 'generic' design that may be used with any sampling situation. Sample size ratios are provided and the user multiplies these times the sample size for a fixed design to obtain the corresponding group sequential analysis times. If delta>0, n.fix is ignored, and delta is taken as the standardized effect size - the signal to noise ratio for a single observation; for example, the mean divided by the standard deviation for a one-sample normal problem. In this case, the sample size at each analysis is computed. When delta=0 and n.fix>1, n.fix is assumed to be the sample size for a fixed design with no interim analyses. See examples below.

Following are further comments on the input argument test.type which is used to control what type of error measurements are used in trial design. The manual may also be worth some review in order to see actual formulas for boundary crossing probabilities for the various options. Options 3 and 5 assume the trial stops if the lower bound is crossed for Type I and Type II error computation (binding lower bound). For the purpose of computing Type I error, options 4 and 6 assume the trial continues if the lower bound is crossed (non-binding lower bound); that is a Type I error can be made by crossing an upper bound after crossing a previous lower bound. Beta-spending refers to error spending for the lower bound crossing probabilities under the alternative hypothesis (options 3 and 4). In this case, the final analysis lower and upper boundaries are assumed to be the same. The appropriate total beta spending (power) is determined by adjusting the maximum sample size through an iterative process for all options. Since options 3 and 4 must compute boundary crossing probabilities under both the null and alternative hypotheses, deriving these designs can take longer than other options. Options 5 and 6 compute lower bound spending under the null hypothesis.

## Usage

```
gsDesign(
    \(\mathrm{k}=3\),
    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 \(=1 \mathrm{e}-06\),
    \(r=18\),
    n. \(\mathrm{I}=0\),
    maxn.IPlan \(=0\),
    nFixSurv = 0,
    endpoint \(=\) NULL,
    delta1 = 1 ,
    delta0 \(=0\),
    overrun = 0,
    usTime = NULL,
    lsTime = NULL
)
\#\# S3 method for class 'gsDesign'
xtable(
    x ,
    caption = NULL,
    label = NULL,
    align = NULL,
    digits = NULL,
    display = NULL,
    ...
)
```


## Arguments

k
Number of analyses planned, including interim and final.
test.type
$1=$ one-sided
2=two-sided symmetric
3=two-sided, asymmetric, beta-spending with binding lower bound
4=two-sided, asymmetric, beta-spending with non-binding lower bound
5=two-sided, asymmetric, lower bound spending under the null hypothesis with binding lower bound
6=two-sided, asymmetric, lower bound spending under the null hypothesis with

|  | non-binding lower bound. <br> See details, examples and manual. |
| :---: | :---: |
| alpha | Type I error, always one-sided. Default value is 0.025. |
| beta | Type II error, default value is 0.1 ( $90 \%$ power). |
| astar | Normally not specified. If test. type=5 or 6 , astar specifies the total probability of crossing a lower bound at all analyses combined. This will be changed to $1-$ alpha when default value of 0 is used. Since this is the expected usage, normally astar is not specified by the user. |
| delta | Effect size for theta under alternative hypothesis. This can be set to the standardized effect size to generate a sample size if n.fix=NULL. See details and examples. |
| n.fix | Sample size for fixed design with no interim; used to find maximum group sequential sample size. For a time-to-event outcome, input number of events required for a fixed design rather than sample size and enter fixed design sample size (optional) in nFixSurv. See details and examples. |
| timing | Sets relative timing of interim analyses. Default of 1 produces equally spaced analyses. Otherwise, this is a vector of length $k$ or $k-1$. The values should satisfy $0<$ timing[1] <timing[2] < . . <timing[k-1] <timing[k]=1. |
| sfu | A spending function or a character string indicating a boundary type (that is, "WT" for Wang-Tsiatis bounds, "OF" for O'Brien-Fleming bounds and "Pocock" for Pocock bounds). For one-sided and symmetric two-sided testing is used to completely specify spending (test.type=1,2), sfu. The default value is sfHSD which is a Hwang-Shih-DeCani spending function. See details, Spending_Function_Overview, manual and examples. |
| sfupar | Real value, default is -4 which is an O'Brien-Fleming-like conservative bound when used with the default Hwang-Shih-DeCani spending function. This is a real-vector for many spending functions. The parameter sfupar specifies any parameters needed for the spending function specified by sfu; this will be ignored for spending functions (sfLDOF, sfLDPocock) or bound types ("OF", "Pocock") that do not require parameters. |
| sfl | Specifies the spending function for lower boundary crossing probabilities when asymmetric, two-sided testing is performed (test. type $=3,4,5$, or 6 ). Unlike the upper bound, only spending functions are used to specify the lower bound. The default value is sfHSD which is a Hwang-Shih-DeCani spending function. The parameter sfl is ignored for one-sided testing (test. type=1) or symmetric 2 -sided testing (test.type=2). See details, spending functions, manual and examples. |
| sflpar | Real value, default is -2 , which, with the default Hwang-Shih-DeCani spending function, specifies a less conservative spending rate than the default for the upper bound. |
| tol | Tolerance for error (default is 0.000001 ). Normally this will not be changed by the user. This does not translate directly to number of digits of accuracy, so use extra decimal places. |
| $r$ | Integer value controlling grid for numerical integration as in Jennison and Turnbull (2000); default is 18 , range is 1 to 80 . Larger values provide larger number of grid points and greater accuracy. Normally $r$ will not be changed by the user. |


| n. I | Used for re-setting bounds when timing of analyses changes from initial design; see examples. |
| :---: | :---: |
| maxn.IPlan | Used for re-setting bounds when timing of analyses changes from initial design; see examples. |
| nFixSurv | If a time-to-event variable is used, nFixSurv computed as the sample size from nSurvival may be entered to have gsDesign compute the total sample size required as well as the number of events at each analysis that will be returned in $n$.fix; this is rounded up to an even number. |
| endpoint | An optional character string that should represent the type of endpoint used for the study. This may be used by output functions. Types most likely to be recognized initially are "TTE" for time-to-event outcomes with fixed design sample size generated by nSurvival() and "Binomial" for 2-sample binomial outcomes with fixed design sample size generated by nBinomial(). |
| delta1 | delta1 and delta0 may be used to store information about the natural parameter scale compared to delta that is a standardized effect size. delta1 is the alternative hypothesis parameter value on the natural parameter scale (e.g., the difference in two binomial rates). |
| delta0 | delta0 is the null hypothesis parameter value on the natural parameter scale. |
| overrun | Scalar or vector of length k-1 with patients enrolled that are not included in each interim analysis. |
| usTime | Default is NULL in which case upper bound spending time is determined by timing. Otherwise, this should be a vector of length $k$ with the spending time at each analysis (see Details). |
| lsTime | Default is NULL in which case lower bound spending time is determined by timing. Otherwise, this should be a vector of length $k$ with the spending time at each analysis (see Details). |
| x | An R object of class found among methods(xtable). See below on how to write additional method functions for xtable. |
| caption | Character vector of length 1 or 2 containing the table's caption or title. If length is 2 , the second item is the "short caption" used when LaTeX generates a "List of Tables". Set to NULL to suppress the caption. Default value is NULL. |
| label | Character vector of length 1 containing the LaTeX label or HTML anchor. Set to NULL to suppress the label. Default value is NULL. |
| align | Character vector of length equal to the number of columns of the resulting table, indicating the alignment of the corresponding columns. Also, " $\mid$ " may be used to produce vertical lines between columns in LaTeX tables, but these are effectively ignored when considering the required length of the supplied vector. If a character vector of length one is supplied, it is split as strsplit(align, "") [[1]] before processing. Since the row names are printed in the first column, the length of align is one greater than $n \operatorname{col}(x)$ if $x$ is a data.frame. Use " $l$ ", " $r$ ", and " $c$ " to denote left, right, and center alignment, respectively. Use " $p\{3 \mathrm{~cm}\}$ " etc. for a LaTeX column of the specified width. For HTML output the " $p$ " alignment is interpreted as " 1 ", ignoring the width request. Default depends on the class of $x$. |


| digits | Numeric vector of length equal to one (in which case it will be replicated as necessary) or to the number of columns of the resulting table or matrix of the same size as the resulting table, indicating the number of digits to display in the corresponding columns. Since the row names are printed in the first column, the length of the vector digits or the number of columns of the matrix digits is one greater than $n \operatorname{col}(x)$ if $x$ is a data.frame. Default depends on the class of $x$. If values of digits are negative, the corresponding values of $x$ are displayed in scientific format with abs(digits) digits. |
| :---: | :---: |
| display | Character vector of length equal to the number of columns of the resulting table, indicating the format for the corresponding columns. Since the row names are printed in the first column, the length of display is one greater than ncol $(x)$ if $x$ is a data.frame. These values are passed to the formatC function. Use "d" (for integers), "f", "e", "E", "g", "G", "fg" (for reals), or "s" (for strings). "f" gives numbers in the usual $x x x . x x x$ format; "e" and "E" give n.ddde+nn or n . dddE +nn (scientific format); " g " and " G " put $\mathrm{x}[\mathrm{i}]$ into scientific format only if it saves space to do so. "fg" uses fixed format as " $f$ ", but digits as number of significant digits. Note that this can lead to quite long result strings. Default depends on the class of $x$. |
|  | Additional arguments. (Currently ignored.) |

## Value

An object of the class gsDesign. This class has the following elements and upon return from gsDesign() contains:

| k | As input. |
| :--- | :--- |
| test.type | As input. |
| alpha |  |
| beta | As input. |
| astar | As input. |
| delta | As input, except when test. type=5 or 6 and astar is input as 0 ; in this case <br> astar is changed to 1-alpha. |
| The standardized effect size for which the design is powered. Will be as input |  |
| to gsDesign() unless it was input as 0; in that case, value will be computed to |  |
| give desired power for fixed design with input sample size n.fix. |  |


| maxn. IPlan | As input. |
| :--- | :--- |
| nFixSurv | As input. |
| nSurv | Sample size for Lachin and Foulkes method when nSurvival is used for fixed <br> design input. If nSurvival is used to compute $n$. fix, then nFixSurv is inflated <br> by the same amount as n.fix and stored in nSurv. Note that if you use gsSurv <br> for time-to-event sample size, this is not needed and a more complete output <br> summary is given. |
| endpoint | As input. |
| delta1 | As input. |
| delta0 | As input. |
| usTime | As input. |
| lsTime | As input. |
| upper | Under bound spending function, boundary and boundary crossing probabilities <br> and manual for further details. |
| lower | Lower bound spending function, boundary and boundary crossing probabilities <br> at each analysis. Lower spending is under alternative hypothesis (beta spend- <br> ing) for test. type=3 or 4. For test. type=2, 5 or 6, lower spending is un- <br> der the null hypothesis. For test. type=1, output value is NULL. See Spend- |
| ing_Function_Overview and manual. |  |

An object of class "xtable" with attributes specifying formatting options for a table

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven \_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall. Lan KK, DeMets DL (1989). Group sequential procedures: calendar versus information time. Statistics in medicine 8(10):1191-8. Liu, Q, Lim, P, Nuamah, I, and Li, Y (2012), On adaptive error spending approach for group sequential trials with random information levels. Journal of biopharmaceutical statistics; 22(4), 687-699.

## See Also

gsDesign package overview, gsBoundSummary, plot.gsDesign, gsProbability, Spending_Function_Overview, Normal xtable

## Examples

```
library(ggplot2)
# symmetric, 2-sided design with O'Brien-Fleming-like boundaries
# lower bound is non-binding (ignored in Type I error computation)
# sample size is computed based on a fixed design requiring n=800
x <- gsDesign(k = 5, test.type = 2, n.fix = 800)
# note that "x" below is equivalent to print(x) and print.gsDesign(x)
x
plot(x)
plot(x, plottype = 2)
# Assuming after trial was designed actual analyses occurred after
# 300, 600, and }860\mathrm{ patients, reset bounds
y <- gsDesign(
    k = 3, test.type = 2, n.fix = 800, n.I = c(300, 600, 860),
    maxn.IPlan = x$n.I[x$k]
)
y
# asymmetric design with user-specified spending that is non-binding
# sample size is computed relative to a fixed design with n=1000
sfup <- c(.033333, .063367, .1)
sflp <- c(.25, .5, .75)
timing <- c(.1, .4, .7)
x <- gsDesign(
    k = 4, timing = timing, sfu = sfPoints, sfupar = sfup, sfl = sfPoints,
    sflpar = sflp, n.fix = 1000
)
x
plot(x)
plot(x, plottype = 2)
# same design, but with relative sample sizes
gsDesign(
    k = 4, timing = timing, sfu = sfPoints, sfupar = sfup, sfl = sfPoints,
    sflpar = sflp
)
```

gsDesign package overview

1.0 Group Sequential Design

## Description

gsDesign is a package for deriving and describing group sequential designs. The package allows particular flexibility for designs with alpha- and beta-spending. Many plots are available for describing design properties.

## Details

Package: gsDesign
Version: 2
License: GPL (version 2 or later)

Index:
gsDesign 2.1: Design Derivation gsProbability 2.2:
Boundary Crossing Probabilities plot.gsDesign 2.3: Plots for group
sequential designs gsCP 2.4: Conditional Power Computation gsBoundCP 2.5:
Conditional Power at Interim Boundaries gsbound 2.6: Boundary derivation -
low level normalGrid 3.1: Normal Density Grid binomial 3.2: Testing,
Confidence Intervals and Sample Size for Comparing Two Binomial Rates
Survival sample size 3.3: Time-to-event sample size calculation
(Lachin-Foulkes) Spending function overview 4.0: Spending functions sfHSD
4.1: Hwang-Shih-DeCani Spending Function sfPower 4.2: Kim-DeMets (power)
Spending Function sfExponential 4.3: Exponential Spending Function
sfLDPocock 4.4: Lan-DeMets Spending function overview sfPoints 4.5:
Pointwise Spending Function sfLogistic 4.6: 2-parameter Spending Function
Families sfTDist 4.7: t-distribution Spending Function Wang-Tsiatis Bounds
5.0: Wang-Tsiatis Bounds checkScalar 6.0: Utility functions to verify
variable properties
The gsDesign package supports group sequential clinical trial design. While there is a strong focus on designs using $\alpha$ - and $\beta$-spending functions, Wang-Tsiatis designs, including O'Brien-Fleming and Pocock designs, are also available. The ability to design with non-binding futility rules allows control of Type I error in a manner acceptable to regulatory authorities when futility bounds are employed.
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. This enables the user to easily create designs with features they desire, such as designs with minimum expected sample size.
Thus, the intent of the gsDesign package is to easily create, fully characterize and even optimize routine group sequential trial designs as well as provide a tool to evaluate innovative designs.

## Author(s)

Keaven Anderson
Maintainer: Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Proschan, MA, Lan, KKG, Wittes, JT (2006), Statistical Monitoring of Clinical Trials. A Unified Approach. New York: Springer.

## See Also

gsDesign, gsProbability

## Examples

```
library(ggplot2)
# assume a fixed design (no interim) trial with the same endpoint
# requires 200 subjects for 90% power at alpha=.025, one-sided
x <- gsDesign(n.fix=200)
plot(x)
```

gsProbability Boundary Crossing Probabilities

## Description

Computes power/Type I error and expected sample size for a group sequential design across a selected set of parameter values for a given set of analyses and boundaries. The print function has been extended using print.gsProbability to print gsProbability objects; see examples.
Depending on the calling sequence, an object of class gsProbability or class gsDesign is returned. If it is of class gsDesign then the members of the object will be the same as described in gsDesign. If $d$ is input as NULL (the default), all other arguments (other than $r$ ) must be specified and an object of class gsProbability is returned. If $d$ is passed as an object of class gsProbability or gsDesign the only other argument required is theta; the object returned has the same class as the input d . On output, the values of theta input to gsProbability will be the parameter values for which the design is characterized.

## Usage

```
gsProbability(k = 0, theta, n.I, a, b, r=18, d = NULL, overrun = 0)
## S3 method for class 'gsProbability'
print(x, ...)
```


## Arguments

k
theta Vector of standardized effect sizes for which boundary crossing probabilities are to be computed.
n. I Sample size or relative sample size at analyses; vector of length k. See gsDesign and manual.
a Lower bound cutoffs (z-values) for futility or harm at each analysis, vector of length k .
b Upper bound cutoffs (z-values) for futility at each analysis; vector of length k .
$r \quad$ Control for grid as in Jennison and Turnbull (2000); default is 18, range is 1 to 80. Normally this will not be changed by the user.
d
If not NULL, this should be an object of type gsDesign returned by a call to gsDesign(). When this is specified, the values of $k$, $n . I, a, b$, and $r$ will be obtained from d and only theta needs to be specified by the user.
overrun Scalar or vector of length k-1 with patients enrolled that are not included in each interim analysis.
$x \quad$ An item of class gsProbability.
... Not implemented (here for compatibility with generic print input).

## Value

k
theta
n. I
lower
upper
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.
$r$ As input.
Note: print.gsProbability() returns the input $x$.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

plot.gsDesign, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# making a gsDesign object first may be easiest...
x <- gsDesign()
# take a look at it
x
# default plot for gsDesign object shows boundaries
plot(x)
# \code{plottype=2} shows boundary crossing probabilities
plot(x, plottype = 2)
# now add boundary crossing probabilities and
# expected sample size for more theta values
y <- gsProbability(d = x, theta = x$delta * seq(0, 2, . 25))
class(y)
# note that "y" below is equivalent to \code{print(y)} and
# \code{print.gsProbability(y)}
y
# the plot does not change from before since this is a
# gsDesign object; note that theta/delta is on x axis
plot(y, plottype = 2)
# now let's see what happens with a gsProbability object
z <- gsProbability(
    k = 3, a = x$lower$bound, b = x$upper$bound,
    n.I = x$n.I, theta = x$delta * seq(0, 2, . 25)
)
# with the above form, the results is a gsProbability object
```

```
    class(z)
    z
    # default plottype is now 2
    # this is the same range for theta, but plot now has theta on x axis
    plot(z)
```

hGraph
Create multiplicity graphs using ggplot 2

## Description

hGraph() plots a multiplicity graph defined by user inputs. The graph can also be used with the $* * * \mathrm{gMCP} * * *$ package to evaluate a set of nominal p -values for the tests of the hypotheses in the graph

## Usage

hGraph (
nHypotheses = 4,
nameHypotheses = paste("H", (1:nHypotheses), sep = ""),
alphaHypotheses $=0.025 / n H y p o t h e s e s$,
$m=$ matrix (array (1/(nHypotheses - 1), nHypotheses^2), nrow = nHypotheses) -
diag(1/(nHypotheses - 1), nHypotheses),
fill = 1 ,
palette = grDevices: :gray.colors(length(unique(fill)), start $=0.5$, end $=0.8$ ),
labels = LETTERS[1:length(unique(fill))],
legend.name = " ",
legend.position = "none",
halfWid $=0.5$,
halfHgt = 0.5,
trhw = 0.1,
trhh $=0.075$,
trprop $=1 / 3$,
digits $=5$,
trdigits $=2$,
size $=6$,
boxtextsize = 4,
arrowsize = 0.02,
radianStart $=$ if ((nHypotheses) $\% \% 2$ ! = 0) \{ pi * (1/2 + $1 / n H y p o t h e s e s)\}$ else $\{$
pi * (1 + 2/nHypotheses)/2 \},
offset $=$ pi/4/nHypotheses,
xradius = 2,
yradius = xradius,
$x=$ NULL,
$y=$ NULL ,
wchar = if (as.character (Sys.info()[1]) == "Windows") \{ "w" \} else \{ "w" \}
)

## Arguments

| nHypotheses | number of hypotheses in graph |
| :---: | :---: |
| nameHypotheses | hypothesis names |
| alphaHypotheses |  |
|  | alpha-levels or weights for ellipses |
| m | square transition matrix of dimension 'nHypotheses' |
| fill | grouping variable for hypotheses |
| palette | colors for groups |
| labels | text labels for groups |
| legend. name | text for legend header |
| legend. position |  |
|  | text string or $\mathrm{x}, \mathrm{y}$ coordinates for legend |
| halfWid | half width of ellipses |
| halfHgt | half height of ellipses |
| trhw | transition box width |
| trhh | transition box height |
| trprop | proportion of transition arrow length where transition box is placed |
| digits | number of digits to show for alphaHypotheses |
| trdigits | digits displayed for transition weights |
| size | text size in ellipses |
| boxtextsize | transition text size |
| arrowsize | size of arrowhead for transition arrows |
| radianStart | radians from origin for first ellipse; nodes spaced equally in clockwise order with centers on an ellipse by default |
| offset | rotational offset in radians for transition weight arrows |
| xradius | horizontal ellipse diameter on which ellipses are drawn |
| yradius | vertical ellipse diameter on which ellipses are drawn |
| x | x coordinates for hypothesis ellipses if elliptical arrangement is not wanted |
| y | y coordinates for hypothesis ellipses if elliptical arrangement is not wanted |
| wchar | character for alphaHypotheses in ellipses |

## Details

See vignette ${ }^{* *}$ Multiplicity graphs formatting using ggplot2** for explanation of formatting.

## Value

A 'ggplot' object with a multi-layer multiplicity graph

## Examples

```
library(tidyr)
# Defaults: note clockwise ordering
hGraph(5)
# Add colors (default is 3 gray shades)
hGraph(3,fill=1:3)
# Colorblind palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
                            "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
hGraph(6,fill=as.factor(1:6),palette=cbPalette)
# Use a hue palette
hGraph(4,fill=factor(1:4),palette=scales::hue_pal(l=75)(4))
# different alpha allocation, hypothesis names and transitions
alphaHypotheses <- c(.005,.007,.013)
nameHypotheses <- c("ORR","PFS","OS")
m <- matrix(c(0,1,0,
                    0,0,1,
    1,0,0),nrow=3,byrow=TRUE)
```

hGraph(3, alphaHypotheses=alphaHypotheses, nameHypotheses=nameHypotheses,m=m)
\# Custom position and size of ellipses, change text to multi-line text
\# Adjust box width
\# add legend in middle of plot
hGraph(3, $x=\operatorname{sqrt}(0: 2), y=c(1,3,1.5)$, size=6, halfWid=.3,halfHgt=.3, trhw=0.6,
palette=cbPalette[2:4], fill = c(1, 2, 2),
legend.position $=c(.6, .5)$, legend. name $=$ "Legend:", labels $=c(" G r o u p 1 ", ~ " G r o u p ~ 2 "), ~$
nameHypotheses=c("H1:\n Long name","H2:\n Longer name","H3: \n Longest name"))
nNormal Normal distribution sample size (2-sample)

## Description

nNormal() computes a fixed design sample size for comparing 2 means where variance is known. T The function allows computation of sample size for a non-inferiority hypothesis. Note that you may wish to investigate other $R$ packages such as the pwr package which uses the $t$-distribution. In the examples below we show how to set up a 2-arm group sequential design with a normal outcome.
nNormal() computes sample size for comparing two normal means when the variance for observations in

## Usage

```
nNormal(
    delta1 = 1,
    sd = 1.7,
    sd2 = NULL,
    alpha = 0.025,
    beta = 0.1,
    ratio = 1,
```

```
    sided = 1,
    n = NULL,
    delta0 = 0,
    outtype = 1
)
```


## Arguments

| delta1 | difference between sample means under the alternate hypothesis. |
| :---: | :---: |
| sd | Standard deviation for the control arm. |
| sd2 | Standard deviation of experimental arm; this will be set to be the same as the control arm with the default of NULL. |
| alpha | type I error rate. Default is 0.025 since 1-sided testing is default. |
| beta | type II error rate. Default is 0.10 ( $90 \%$ power). Not needed if n is provided. |
| ratio | randomization ratio of experimental group compared to control. |
| sided | 1 for 1 -sided test (default), 2 for 2 -sided test. |
| n | Sample size; may be input to compute power rather than sample size. If NULL (default) then sample size is computed. |
| delta0 | difference between sample means under the null hypothesis; normally this will be left as the default of 0 . |
| outtype | controls output; see value section below. |

## Details

This is more of a convenience routine than one recommended for broad use without careful considerations such as those outlined in Jennison and Turnbull (2000). For larger studies where a conservative estimate of within group standard deviations is available, it can be useful. A more detailed formulation is available in the vignette on two-sample normal sample size.

## Value

If n is NULL (default), total sample size ( 2 arms combined) is computed. Otherwise, power is computed. If outtype $=1$ (default), the computed value (sample size or power) is returned in a scalar or vector. If outtype=2, a data frame with sample sizes for each arm ( $n 1, n 2$ ) is returned; if $n$ is not input as NULL, a third variable, Power, is added to the output data frame. If outtype=3, a data frame with is returned with the following columns:
$\mathrm{n} \quad$ A vector with total samples size required for each event rate comparison specified
n1 A vector of sample sizes for group 1 for each event rate comparison specified
n2 A vector of sample sizes for group 2 for each event rate comparison specified
alpha As input
sided As input
beta As input; if n is input, this is computed

| Power | If $n=$ NULL on input, this is 1-beta; otherwise, the power is computed for each <br> sample size input |
| :--- | :--- |
| sd | As input |
| sd2 | As input |
| delta1 | As input |
| delta 0 | As input |
| se | standard error for estimate of difference in treatment group means |

## Author(s)

Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

Lachin JM (1981), Introduction to sample size determination and power analysis for clinical trials. Controlled Clinical Trials 2:93-113.

Snedecor GW and Cochran WG (1989), Statistical Methods. 8th ed. Ames, IA: Iowa State University Press.

## See Also

gsDesign package overview

## Examples

```
# EXAMPLES
# equal variances
n=nNormal(delta1=.5,sd=1.1,alpha=.025,beta=.2)
n
x <- gsDesign(k = 3, n.fix = n, test.type = 4, alpha = 0.025, beta = 0.1, timing = c(.5,.75),
sfu = sfLDOF, sfl = sfHSD, sflpar = -1, delta1 = 0.5, endpoint = 'normal')
gsBoundSummary(x)
summary (x)
# unequal variances, fixed design
nNormal(delta1 = .5, sd = 1.1, sd2 = 2, alpha = .025, beta = .2)
# unequal sample sizes
nNormal(delta1 = .5, sd = 1.1, alpha = .025, beta = . 2, ratio = 2)
# non-inferiority assuming a better effect than null
nNormal(delta1 = .5, delta0 = -.1, sd = 1.2)
```


## Description

normalGrid() is intended to be used for computation of the expected value of a function of a normal random variable. The function produces grid points and weights to be used for numerical integration.

This is a utility function to provide a normal density function and a grid to integrate over as described by Jennison and Turnbull (2000), Chapter 19. While integration can be performed over the real line or over any portion of it, the numerical integration does not extend beyond 6 standard deviations from the mean. The grid used for integration uses equally spaced points over the middle of the distribution function, and spreads points further apart in the tails. The values returned in gridwgts may be used to integrate any function over the given grid, although the user should take care that the function integrated is not large in the tails of the grid where points are spread further apart.

## Usage

normalGrid( $r=18$, bounds $=c(0,0), m u=0$, sigma $=1)$

## Arguments

$r \quad$ Control for grid points as in Jennison and Turnbull (2000), Chapter 19; default is 18 . Range: 1 to 80 . This might be changed by the user (e.g., $r=6$ which produces 65 gridpoints compare to 185 points when $r=18$ ) when speed is more important than precision.
bounds Range of integration. Real-valued vector of length 2. Default value of 0,0 produces a range of + or -6 standard deviations ( $6 *$ sigma) from the mean $(=m u)$.
mu Mean of the desired normal distribution.
sigma Standard deviation of the desired normal distribution.

## Value

Z
density The standard normal density function evaluated at the values in $z$; see examples.
gridwgts Simpson's rule weights for numerical integration on the grid in z ; see examples.
wgts Weights to be used with the grid in $z$ for integrating the normal density function; see examples. This is equal to density * gridwgts.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.com>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## Examples

```
library(ggplot2)
# standard normal distribution
x <- normalGrid(r = 3)
plot(x$z, x$wgts)
# verify that numerical integration replicates sigma
# get grid points and weights
x <- normalGrid(mu = 2, sigma = 3)
# compute squared deviation from mean for grid points
dev <- (x$z - 2)^2
# multiply squared deviations by integration weights and sum
sigma2 <- sum(dev * x$wgts)
# square root of sigma2 should be sigma (3)
sqrt(sigma2)
# do it again with larger r to increase accuracy
x <- normalGrid(r = 22, mu = 2, sigma = 3)
sqrt(sum((x$z - 2)^2 * x$wgts))
# this can also be done by combining gridwgts and density
sqrt(sum((x$z - 2)^2 * x$gridwgts * x$density))
# integrate normal density and compare to built-in function
# to compute probability of being within 1 standard deviation
# of the mean
pnorm(1) - pnorm(-1)
x <- normalGrid(bounds = c(-1, 1))
sum(x$wgts)
sum(x$gridwgts * x$density)
# find expected sample size for default design with
# n.fix=1000
x <- gsDesign(n.fix = 1000)
x
# set a prior distribution for theta
y <- normalGrid(r = 3, mu = x$theta[2], sigma = x$theta[2] / 1.5)
z <- gsProbability(
    k = 3, theta = y$z, n.I = x$n.I, a = x$lower$bound,
```

```
    b = x$upper$bound
)
z <- gsProbability(d = x, theta = y$z)
cat(
    "Expected sample size averaged over normal\n prior distribution for theta with \n mu=",
    x$theta[2], "sigma=", x$theta[2] / 1.5, ":",
    round(sum(z$en * y$wgt), 1), "\n"
)
plot(y$z, z$en,
    xlab = "theta", ylab = "E{N}",
    main = "Expected sample size for different theta values"
)
lines(y$z, z$en)
```

```
plot.gsDesign Plots for group sequential designs
```


## Description

The plot() function has been extended to work with objects returned by gsDesign() and gsProbability(). For objects of type gsDesign, seven types of plots are provided: z-values at boundaries (default), power, approximate treatment effects at boundaries, conditional power at boundaries, spending functions, expected sample size, and B-values at boundaries. For objects of type gsProbability plots are available for z -values at boundaries, power (default), approximate treatment effects at boundaries, conditional power, expected sample size and B-values at boundaries.

The intent is that many standard plot() parameters will function as expected; exceptions to this rule exist. In particular, main, xlab, ylab, lty , col , lwd, type, pch, cex have been tested and work for most values of plottype; one exception is that type="l" cannot be overridden when plottype=2. Default values for labels depend on plottype and the class of $x$.

Note that there is some special behavior for values plotted and returned for power and expected sample size (ASN) plots for a gsDesign object. A call to $\mathrm{x}<-\mathrm{gsDesign}()$ produces power and expected sample size for only two theta values: 0 and $x \$ d e l t a$. The call plot ( $x, p l o t t y p e=" P o w e r "$ ) (or plot ( $\mathrm{x}, \mathrm{plottype=}$ "ASN") for a gsDesign object produces power (expected sample size) curves and returns a gsDesign object with theta values determined as follows. If theta is non-null on input, the input value(s) are used. Otherwise, for a gsProbability object, the theta values from that object are used. For a gsDesign object where theta is input as NULL (the default), theta=seq $(0,2, .05) * x \$ d e l t a)$ is used. For a gsDesign object, the $x$-axis values are rescaled to theta/x\$delta and the label for the x-axis theta/delta. For a gsProbability object, the values of theta are plotted and are labeled as theta. See examples below.

Approximate treatment effects at boundaries are computed dividing the Z-values at the boundaries by the square root of $n$. I at that analysis.

Spending functions are plotted for a continuous set of values from 0 to 1 . This option should not be used if a boundary is used or a pointwise spending function is used (sfu or sfl="WT", "OF", "Pocock" or sfPoints).

Conditional power is computed using the function gsBoundCP(). The default input for this routine is theta="thetahat" which will compute the conditional power at each bound using the approximate treatment effect at that bound. Otherwise, if the input is gsDesign object conditional power is computed assuming theta $=x \$ d e l t a$, the original effect size for which the trial was planned.
Average sample number/expected sample size is computed using n. I at each analysis times the probability of crossing a boundary at that analysis. If no boundary is crossed at any analysis, this is counted as stopping at the final analysis.
$B$-values are Z-values multiplied by sqrt $(t)=\operatorname{sqrt}(x \$ n . I / x \$ n . I[x \$ k])$. Thus, the expected value of a B-value at an analysis is the true value of theta multiplied by the proportion of total planned observations at that time. See Proschan, Lan and Wittes (2006).

## Usage

```
## S3 method for class 'gsDesign'
plot(x, plottype = 1, base = FALSE, ...)
## S3 method for class 'gsProbability'
plot(x, plottype = 2, base = FALSE, ...)
```


## Arguments

x
plottype $\quad 1=$ boundary plot (default for gsDesign),
2=power plot (default for gsProbability),
3=approximate treatment effect at boundaries, $4=$ conditional power at boundaries, $5=$ spending function plot (only available if class $(x)==" g s D e s i g n "$ ),
$6=$ expected sample size plot, and
$7=B$-values at boundaries.
Character values for plottype may also be entered: "Z" for plot type 1 , "power" for plot type 2, "thetahat" for plot type 3, "CP" for plot type 4, "sf" for plot type 5 , "ASN", " $N$ " or " $n$ " for plot type 6 , and " $B$ ", " $B-v a l$ " or " $B$-value" for plot type 7.
base Default is FALSE, which means ggplot2 graphics are used. If true, base graphics are used for plotting.
This allows many optional arguments that are standard when calling plot.
Other arguments include:
theta which is used for plottype $=2,4,6$; normally defaults will be adequate; see details.
ses=TRUE which applies only when plottype=3 and
class(x)=="gsDesign"; indicates that approximate standardized effect size at the boundary is to be plotted rather than the approximate natural parameter.
xval="Default" which is only effective when plottype=2 or 6. Appropriately scaled (reparameterized) values for x -axis for power and expected sample size graphs; see details.

## Value

An object of class( $x$ ); in many cases this is the input value of $x$, while in others $x \$$ theta is replaced and corresponding characteristics computed; see details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.com>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Proschan, MA, Lan, KKG, Wittes, JT (2006), Statistical Monitoring of Clinical Trials. A Unified Approach. New York: Springer.

## See Also

gsDesign, gsProbability

## Examples

```
library(ggplot2)
# symmetric, 2-sided design with O'Brien-Fleming-like boundaries
# lower bound is non-binding (ignored in Type I error computation)
# sample size is computed based on a fixed design requiring n=100
x <- gsDesign(k = 5, test.type = 2, n.fix = 100)
x
# the following translate to calls to plot.gsDesign since x was
# returned by gsDesign; run these commands one at a time
plot(x)
plot(x, plottype = 2)
plot(x, plottype = 3)
plot(x, plottype = 4)
plot(x, plottype = 5)
plot(x, plottype = 6)
plot(x, plottype = 7)
# choose different parameter values for power plot
# start with design in x from above
y <- gsProbability(
    k = 5, theta = seq(0, . 5, .025), x$n.I,
    x$lower$bound, x$upper$bound
)
# the following translates to a call to plot.gsProbability since
```

```
\# y has that type
```

plot (y)
print.nSurv Advanced time-to-event sample size calculation

## Description

nSurv () is used to calculate the sample size for a clinical trial with a time-to-event endpoint and an assumption of proportional hazards. This set of routines is new with version 2.7 and will continue to be modified and refined to improve input error checking and output format with subsequent versions. It allows both the Lachin and Foulkes (1986) method (fixed trial duration) as well as the Kim and Tsiatis(1990) method (fixed enrollment rates and either fixed enrollment duration or fixed minimum follow-up). Piecewise exponential survival is supported as well as piecewise constant enrollment and dropout rates. The methods are for a 2 -arm trial with treatment groups referred to as experimental and control. A stratified population is allowed as in Lachin and Foulkes (1986); this method has been extended to derive non-inferiority as well as superiority trials. Stratification also allows power calculation for meta-analyses. gsSurv() combines nSurv() with gsDesign() to derive a group sequential design for a study with a time-to-event endpoint.

```
Usage
    \#\# S3 method for class 'nSurv'
    print(x, digits \(=4, \ldots\) )
    nSurv(
    lambdaC \(=\log (2) / 6\),
    \(\mathrm{hr}=0.6\),
    hr0 = 1,
    eta \(=0\),
    etaE = NULL,
    gamma = 1,
    \(R=12\),
    S = NULL,
    T = NULL,
    minfup = NULL,
    ratio \(=1\),
    alpha \(=0.025\),
    beta \(=0.1\),
    sided = 1,
    tol \(=\). Machine\$double. eps \(^{\wedge} 0.25\)
)
tEventsIA(x, timing = 0.25, tol = .Machine\$double.eps^0.25)
nEventsIA(tIA \(=5, x=\) NULL, target \(=0\), simple \(=\) TRUE)
```

```
gsSurv(
    k = 3,
    test.type = 4,
    alpha = 0.025,
    sided = 1,
    beta = 0.1,
    astar = 0,
    timing = 1,
    sfu = sfHSD,
    sfupar = -4,
    sfl = sfHSD,
    sflpar = -2,
    r = 18,
    lambdaC = log(2)/6,
    hr = 0.6,
    hr0 = 1,
    eta = 0,
    etaE = NULL,
    gamma = 1,
    R = 12,
    S = NULL,
    T = NULL,
    minfup = NULL,
    ratio = 1,
    tol = .Machine$double.eps^0.25,
    usTime = NULL,
    lsTime = NULL
)
## S3 method for class 'gsSurv'
print(x, digits = 2, ...)
## S3 method for class 'gsSurv'
xtable(
    x,
    caption = NULL,
    label = NULL,
    align = NULL,
    digits = NULL,
    display = NULL,
    auto = FALSE,
    footnote = NULL,
    fnwid = "9cm",
    timename = "months",
)
```


## Arguments

X
digits Number of digits past the decimal place to print (print.gsSurv.); also a pass
An object of class nSurv or gsSurv. print.nSurv() is used for an object of class nSurv which will generally be output from nSurv(). For print.gsSurv() is used for an object of class gsSurv which will generally be output from gsSurv(). nEventsIA and tEventsIA operate on both the nSurv and gsSurv class.
through to generic xtable() from xtable.gsSurv(). other arguments that may be passed to generic functions underlying the methods here.
scalar, vector or matrix of event hazard rates for the control group; rows represent time periods while columns represent strata; a vector implies a single stratum.
$\mathrm{hr} \quad$ hazard ratio (experimental/control) under the alternate hypothesis (scalar).
hr0 hazard ratio (experimental/control) under the null hypothesis (scalar).
eta scalar, vector or matrix of dropout hazard rates for the control group; rows represent time periods while columns represent strata; if entered as a scalar, rate is constant across strata and time periods; if entered as a vector, rates are constant across strata.
matrix dropout hazard rates for the experimental group specified in like form as eta; if NULL, this is set equal to eta.
gamma

R

S

T
minfup
ratio
alpha beta
sided
tol type I error rate. Default is 0.025 since 1 -sided testing is default. type II error rate. Default is 0.10 ( $90 \%$ power); NULL if power is to be computed based on other input values.
1 for 1 -sided testing, 2 for 2 -sided testing.
for cases when T or minfup values are derived through root finding (T or minfup input as NULL), tol provides the level of error input to the uniroot() rootfinding function. The default is the same as for uniroot.

| timing | Sets relative timing of interim analyses in gsSurv. Default of 1 produces equally spaced analyses. Otherwise, this is a vector of length $k$ or $k-1$. The values should satisfy $0<$ timing[1] < timing[2] < . . < timing[k-1] <timing[k]=1. For tEventsIA, this is a scalar strictly between 0 and 1 that indicates the targeted proportion of final planned events available at an interim analysis. |
| :---: | :---: |
| tIA | Timing of an interim analysis; should be between 0 and $\mathrm{y} \$ \mathrm{~T}$. |
| target | The targeted proportion of events at an interim analysis. This is used for rootfinding will be 0 for normal use. |
| simple | See output specification for $n$ EventsIA(). |
| k | Number of analyses planned, including interim and final. |
| test.type | 1=one-sided <br> 2=two-sided symmetric <br> $3=$ two-sided, asymmetric, beta-spending with binding lower bound <br> 4=two-sided, asymmetric, beta-spending with non-binding lower bound <br> 5=two-sided, asymmetric, lower bound spending under the null hypothesis with <br> binding lower bound <br> 6=two-sided, asymmetric, lower bound spending under the null hypothesis with non-binding lower bound. <br> See details, examples and manual. |
| astar | Normally not specified. If test.type=5 or 6, astar specifies the total probability of crossing a lower bound at all analyses combined. This will be changed to 1 -alpha when default value of 0 is used. Since this is the expected usage, normally astar is not specified by the user. |
| sfu | A spending function or a character string indicating a boundary type (that is, "WT" for Wang-Tsiatis bounds, "OF" for O'Brien-Fleming bounds and "Pocock" for Pocock bounds). For one-sided and symmetric two-sided testing is used to completely specify spending (test.type=1,2), sfu. The default value is sfHSD which is a Hwang-Shih-DeCani spending function. See details, Spending_Function_Overview, manual and examples. |
| sfupar | Real value, default is -4 which is an O'Brien-Fleming-like conservative bound when used with the default Hwang-Shih-DeCani spending function. This is a real-vector for many spending functions. The parameter sfupar specifies any parameters needed for the spending function specified by sfu; this will be ignored for spending functions (sfLDOF, sfLDPocock) or bound types ("OF", "Pocock") that do not require parameters. |
| sfl | Specifies the spending function for lower boundary crossing probabilities when asymmetric, two-sided testing is performed (test.type $=3,4,5$, or 6 ). Unlike the upper bound, only spending functions are used to specify the lower bound. The default value is sfHSD which is a Hwang-Shih-DeCani spending function. The parameter sfl is ignored for one-sided testing (test. type=1) or symmetric 2-sided testing (test.type=2). See details, spending functions, manual and examples. |
| sflpar | Real value, default is -2 , which, with the default Hwang-Shih-DeCani spending function, specifies a less conservative spending rate than the default for the upper bound. |


| $r$ | Integer value controlling grid for numerical integration as in Jennison and Turn- <br> bull (2000); default is 18, range is 1 to 80 . Larger values provide larger number <br> of grid points and greater accuracy. Normally $r$ will not be changed by the user. <br> Default is NULL in which case upper bound spending time is determined by <br> timing. Otherwise, this should be a vector of length codek with the spending <br> time at each analysis (see Details in help for gsDesign). <br> Default is NULL in which case lower bound spending time is determined by <br> timing. Otherwise, this should be a vector of length k with the spending time <br> at each analysis (see Details in help for gsDesign). |
| :--- | :--- |
| lsTime | passed through to generic xtable(). |
| caption | passed through to generic xtable(). <br> passed through to generic xtable(). <br> align |
| display | passed through to generic xtable(). <br> passed through to generic xtable(). |
| auto | footnote for xtable output; may be useful for describing some of the design <br> farameters. |
| fnwid | a text string controlling the width of footnote text at the bottom of the xtable |
| timename | output. <br> character string with plural of time units (e.g., "months") |

## Details

print(), xtable() and summary() methods are provided to operate on the returned value from gsSurv(), an object of class gsSurv. print() is also extended to nSurv objects. The functions gsBoundSummary (data frame for tabular output), xprint (application of xtable for tabular output) and summary.gsSurv (textual summary of gsDesign or gsSurv object) may be preferred summary functions; see example in vignettes. See also gsBoundSummary for output of tabular summaries of bounds for designs produced by gsSurv().
Both nEventsIA and tEventsIA require a group sequential design for a time-to-event endpoint of class gsSurv as input. nEventsIA calculates the expected number of events under the alternate hypothesis at a given interim time. tEventsIA calculates the time that the expected number of events under the alternate hypothesis is a given proportion of the total events planned for the final analysis.
nSurv () produces an object of class nSurv with the number of subjects and events for a set of prespecified trial parameters, such as accrual duration and follow-up period. The underlying power calculation is based on Lachin and Foulkes (1986) method for proportional hazards assuming a fixed underlying hazard ratio between 2 treatment groups. The method has been extended here to enable designs to test non-inferiority. Piecewise constant enrollment and failure rates are assumed and a stratified population is allowed. See also nSurvival for other Lachin and Foulkes (1986) methods assuming a constant hazard difference or exponential enrollment rate.

When study duration ( T ) and follow-up duration (minfup) are fixed, nSurv applies exactly the Lachin and Foulkes (1986) method of computing sample size under the proportional hazards assumption when For this computation, enrollment rates are altered proportionately to those input in gamma to achieve the power of interest.

Given the specified enrollment rate(s) input in gamma, nSurv may also be used to derive enrollment duration required for a trial to have defined power if $T$ is input as NULL; in this case, both $R$ (enrollment duration for each specified enrollment rate) and T (study duration) will be computed on output.
Alternatively and also using the fixed enrollment rate(s) in gamma, if minimum follow-up minfup is specified as NULL, then the enrollment duration(s) specified in $R$ are considered fixed and minfup and T are computed to derive the desired power. This method will fail if the specified enrollment rates and durations either over-powers the trial with no additional follow-up or underpowers the trial with infinite follow-up. This method produces a corresponding error message in such cases.
The input to gsSurv is a combination of the input to $n S u r v()$ and gsDesign().
nEventsIA() is provided to compute the expected number of events at a given point in time given enrollment, event and censoring rates. The routine is used with a root finding routine to approximate the approximate timing of an interim analysis. It is also used to extend enrollment or follow-up of a fixed design to obtain a sufficient number of events to power a group sequential design.

## Value

nSurv() returns an object of type nSurv with the following components:

| alpha | As input. |
| :---: | :---: |
| sided | As input. |
| beta | Type II error; if missing, this is computed. |
| power | Power corresponding to input beta or computed if output beta is computed. |
| lambdaC | As input. |
| etaC | As input. |
| etaE | As input. |
| gamma | As input unless none of the following are NULL: T, minfup, beta; otherwise, this is a constant times the input value required to power the trial given the other input variables. |
| ratio | As input. |
| R | As input unless T was NULL on input. |
| S | As input. |
| T | As input. |
| minfup | As input. |
| hr | As input. |
| hr0 | As input. |
| n | Total expected sample size corresponding to output accrual rates and durations. |
| d | Total expected number of events under the alternate hypothesis. |
| tol | As input, except when not used in computations in which case this is returned as NULL. This and the remaining output below are not printed by the print() extension for the nSurv class. |
| eDC | A vector of expected number of events by stratum in the control group under the alternate hypothesis. |


| eDE | A vector of expected number of events by stratum in the experimental group <br> under the alternate hypothesis. |
| :--- | :--- |
| eDC0 | A vector of expected number of events by stratum in the control group under the <br> null hypothesis. |
| eDE0 | A vector of expected number of events by stratum in the experimental group <br> under the null hypothesis. |
| eNC | A vector of the expected accrual in each stratum in the control group. |
| eNE | A vector of the expected accrual in each stratum in the experimental group. |
| variable | A text string equal to "Accrual rate" if a design was derived by varying the <br> accrual rate, "Accrual duration" if a design was derived by varying the accrual <br> duration, "Follow-up duration" if a design was derived by varying follow-up <br> duration, or "Power" if accrual rates and duration as well as follow-up duration <br> was specified and beta=NULL was input. |

gsSurv() returns much of the above plus an object of class gsDesign in a variable named gs; see gsDesign for general documentation on what is returned in gs. The value of gs $\$ n$. I represents the number of endpoints required at each analysis to adequately power the trial. Other items returned by gsSurv() are:

| gs | A group sequential design (gsDesign) object. |
| :---: | :---: |
| lambdaC | As input. |
| etaC | As input. |
| etaE | As input. |
| gamma | As input unless none of the following are NULL: T, minfup, beta; otherwise, this is a constant times the input value required to power the trial given the other input variables. |
| ratio | As input. |
| R | As input unless T was NULL on input. |
| S | As input. |
| T | As input. |
| minfup | As input. |
| hr | As input. |
| hr0 | As input. |
| eNC | Total expected sample size corresponding to output accrual rates and durations. |
| eNE | Total expected sample size corresponding to output accrual rates and durations. |
| eDC | Total expected number of events under the alternate hypothesis. |
| eDE | Total expected number of events under the alternate hypothesis. |
| tol | As input, except when not used in computations in which case this is returned as NULL. This and the remaining output below are not printed by the print() extension for the nSurv class. |
| eDC | A vector of expected number of events by stratum in the control group under the alternate hypothesis. |

eDE | A vector of expected number of events by stratum in the experimental group |
| :--- |
| under the alternate hypothesis. |
| A vector of expected number of events by stratum in the control group under the |
| null hypothesis. |

eDE0 | A vector of expected number of events by stratum in the experimental group |
| :--- |
| under the null hypothesis. |

eNC A vector of the expected accrual in each stratum in the control group.
eNE

## Author(s)

Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)

## References

Kim KM and Tsiatis AA (1990), Study duration for clinical trials with survival response and early stopping rule. Biometrics, 46, 81-92
Lachin JM and Foulkes MA (1986), 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, 507-519.

Schoenfeld D (1981), The Asymptotic Properties of Nonparametric Tests for Comparing Survival Distributions. Biometrika, 68, 316-319.

## See Also

gsBoundSummary, xprint, gsDesign package overview, plot.gsDesign, gsDesign, gsHR, nSurvival uniroot

Normal xtable

## Examples

```
# vary accrual rate to obtain power
nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 1, T = 36, minfup = 12)
# vary accrual duration to obtain power
nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 6, minfup = 12)
# vary follow-up duration to obtain power
nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 6, R = 25)
# piecewise constant enrollment rates (vary accrual duration)
nSurv(
    lambdaC = log(2) / 6, hr = . 5, eta = log(2) / 40, gamma = c(1, 3, 6),
    R = c(3, 6, 9), minfup = 12
)
# stratified population (vary accrual duration)
nSurv(
    lambdaC = matrix(log(2) / c(6, 12), ncol = 2), hr = .5, eta = log(2) / 40,
    gamma = matrix(c(2, 4), ncol = 2), minfup = 12
)
# piecewise exponential failure rates (vary accrual duration)
nSurv(lambdaC = log(2)/c(6, 12), hr = . 5, eta = log(2) / 40, S = 3, gamma = 6, minfup = 12)
# combine it all: 2 strata, 2 failure rate periods
nSurv(
    lambdaC = matrix(log(2) / c(6, 12, 18, 24), ncol = 2), hr = .5,
    eta = matrix(log(2) / c(40, 50, 45, 55), ncol = 2), S = 3,
    gamma = matrix(c(3, 6, 5, 7), ncol = 2), R = c(5, 10), minfup = 12
)
# example where only 1 month of follow-up is desired
# set failure rate to 0 after 1 month using lambdaC and S
nSurv(lambdaC = c(.4, 0), hr = 2 / 3, S = 1, minfup = 1)
# group sequential design (vary accrual rate to obtain power)
x <- gsSurv(
    k = 4, sfl = sfPower, sflpar = .5, lambdaC = log(2) / 6, hr = .5,
    eta = log(2) / 40, gamma = 1, T = 36, minfup = 12
)
x
print(xtable::xtable(x,
    footnote = "This is a footnote; note that it can be wide.",
    caption = "Caption example."
))
# find expected number of events at time 12 in the above trial
nEventsIA(x = x, tIA = 10)
# find time at which 1/4 of events are expected
tEventsIA(x = x, timing = .25)
```

print.nSurvival Time-to-event sample size calculation(Lachin-Foulkes)

## Description

nSurvival() is used to calculate the sample size for a clinical trial with a time-to-event endpoint. The Lachin and Foulkes (1986) method is used. nEvents uses the Schoenfeld (1981) approximation to provide sample size and power in terms of the underlying hazard ratio and the number of events observed in a survival analysis. The functions $\operatorname{hrz2n(),hrn2z()~and~zn2hr()~also~use~the~}$ Schoenfeld approximation to provide simple translations between hazard ratios, $z$-values and the number of events in an analysis; input variables can be given as vectors.
nSurvival() produces an object of class "nSurvival" with the number of subjects and events for a set of pre-specified trial parameters, such as accrual duration and follow-up period. The calculation is based on Lachin and Foulkes (1986) method and can be used for risk ratio or risk difference. The function also consider non-uniform (exponential) entry as well as uniform entry.
If the logical approx is TRUE, the variance under alternative hypothesis is used to replace the variance under null hypothesis. For non-uniform entry, a non-zero value of gamma for exponential entry must be supplied. For positive gamma, the entry distribution is convex, whereas for negative gamma, the entry distribution is concave.
nEvents() uses the Schoenfeld (1981) method to approximate the number of events $n$ (given beta) or the power (given n). Arguments may be vectors or scalars, but any vectors must have the same length.

The functions hrz2n, hrn2z and zn2hr also all apply the Schoenfeld approximation for proportional hazards modeling. This approximation is based on the asymptotic normal distribtuion of the logrank statistic as well as related statistics are asymptotically normal. Let $\lambda$ denote the underlying hazard ratio (lambda1/lambda2 in terms of the arguments to nSurvival). Further, let $n$ denote the number of events observed when computing the statistic of interest and $r$ the ratio of the sample size in an experimental group relative to a control. The estimated natural logarithm of the hazard ratio from a proportional hazards ratio is approximately normal with a mean of $\log \lambda$ and variance $(1+$ $r)^{2} / n r$. Let $z$ denote a logrank statistic (or a Wald statistic or score statistic from a proportional hazards regression model). The same asymptotic theory implies $z$ is asymptotically equivalent to a normalized estimate of the hazard ratio $\lambda$ and thus $z$ is asymptotically normal with variance 1 and mean

$$
\frac{\log \lambda r}{(1+r)^{2}}
$$

Plugging the estimated hazard ratio into the above equation allows approximating any one of the following based on the other two: the estimate hazard ratio, the number of events and the z -statistic. That is,

$$
\begin{gathered}
\hat{\lambda}=\exp (z(1+r) / \sqrt{r n}) \\
z=\log (\hat{\lambda}) \sqrt{n r} /(1+r) \\
n=(z(1+r) / \log (\hat{\lambda}))^{2} / r
\end{gathered}
$$

$\mathrm{hrz2n}$ () translates an observed interim hazard ratio and interim z -value into the number of events required for the Z-value and hazard ratio to correspond to each other. hrn2z() translates a hazard ratio and number of events into an approximate corresponding Z-value. zn2hr() translates a

Z-value and number of events into an approximate corresponding hazard ratio. Each of these functions has a default assumption of an underlying hazard ratio of 1 which can be changed using the argument hr 0 . hrn 2 z () and zn 2 hr() also have an argument hr 1 which is only used to compute the sign of the computed Z-value in the case of hrn 2 z() and whether or not a z -value $>0$ corresponds to a hazard ratio $>$ or $<$ the null hazard ratio hr 0 .

## Usage

```
## S3 method for class 'nSurvival'
print(x, ...)
nSurvival(
    lambda1 = 1/12,
    lambda2 = 1/24,
    Ts = 24,
    Tr = 12,
    eta = 0,
    ratio = 1,
    alpha = 0.025,
    beta = 0.1,
    sided = 1,
    approx = FALSE,
    type = c("rr", "rd"),
    entry = c("unif", "expo"),
    gamma = NA
)
nEvents(
    hr = 0.6,
    alpha = 0.025,
    beta = 0.1,
    ratio = 1,
    sided = 1,
    hr0 = 1,
    n = 0,
    tbl = FALSE
)
    zn2hr(z, n, ratio = 1, hr0 = 1, hr1 = 0.7)
    hrn2z(hr, n, ratio = 1, hr0 = 1, hr1 = 0.7)
    hrz2n(hr, z, ratio = 1, hr0 = 1)
```


## Arguments

x
An object of class "nSurvival" returned by nSurvival() (optional: used for output; "months" or "years" would be the 'usual' choices).
... Allows additional arguments for print.nSurvival().

```
lambda1, lambda2
```

event hazard rate for placebo and treatment group respectively.
Ts maximum study duration.
Tr accrual (recruitment) duration.
eta equal dropout hazard rate for both groups.
ratio randomization ratio between placebo and treatment group. Default is balanced
design, i.e., randomization ratio is 1 .
alpha type I error rate. Default is 0.025 since 1 -sided testing is default.
beta type II error rate. Default is 0.10 ( $90 \%$ power). Not needed for $n E v e n t s()$ if $n$
is provided.
sided one or two-sided test? Default is one-sided test.
approx logical. If TRUE, the approximation sample size formula for risk difference is
used.
type type of sample size calculation: risk ratio ("rr") or risk difference ("rd").
entry patient entry type: uniform entry ("unif") or exponential entry ("expo").
gamma rate parameter for exponential entry. NA if entry type is "unif" (uniform). A
non-zero value is supplied if entry type is "expo" (exponential).
hr Hazard ratio. For nEvents, this is the hazard ratio under the alternative hypoth-
esis ( $>0$ ).
hr0 Hazard ratio under the null hypothesis ( $>0$, for nEvents, $!=h r$ ).
$\mathrm{n} \quad$ Number of events. For nEvents may be input to compute power rather than
sample size.
tbl Indicator of whether or not scalar (vector) or tabular output is desired for nEvents().
z
A z-statistic.
$\mathrm{hr} 1 \quad$ Hazard ratio under the alternate hypothesis for $\mathrm{hrn} 2 \mathrm{z}, \mathrm{zn} 2 \mathrm{hr}(>0,!=\mathrm{hr} 0)$

## Value

nSurvival produces a list with the following component returned:

| type | As input. |
| :--- | :--- |
| entry | As input. |


| n | Sample size required (computed). |
| :--- | :--- |
| nEvents | Number of events required (computed). |

lambda1 As input.
lambda2 As input.
eta As input.
ratio As input.
gamma As input.
alpha As input.
beta As input.

| sided | As input. |
| :--- | :--- |
| Ts | As input. |
| Tr | As input. |

nEvents produces a scalar or vector of sample sizes (or powers) when tbl=FALSE or, when tbl=TRUE a data frame of values with the following columns:
hr As input
$n \quad$ If $n[1]=0$ on input (default), output contains the number of events need to obtain the input Type I and II error. If $n[1]>0$ on input, the input value is returned.
alpha As input.
beta If $n[1]=0$ on input (default), beta is output as input. Otherwise, this is the computed Type II error based on the input $n$.

Power One minus the output beta. When tbl=FALSE, $\mathrm{n}[1]>0$, this is the value or vector of values returned.
delta Standardized effect size represented by input difference between null and alternative hypothesis hazard ratios.
ratio Ratio of experimental to control sample size where 'experimental' is the same as the group with hazard represented in the numerator of the hazard ratio.
se Estimated standard error for the observed $\log$ (hazard ratio) with the given sample size.
$h r z 2 n$ outputs a number of events required to approximately have the input hazard ratio, $z$-statistic and sample size correspond. hrn2z outputs an approximate z-statistic corresponding to an input hazard ratio and number of events. zn2hr outputs an approximate hazard ratio corresponding to an input z -statistic and number of events.

## Author(s)

Shanhong Guan [shanhong.guan@gmail.com](mailto:shanhong.guan@gmail.com), Keaven Anderson [keaven_anderson@merck.com](mailto:keaven_anderson@merck.com)

## References

Lachin JM and Foulkes MA (1986), 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, 507-519.

Schoenfeld D (1981), The Asymptotic Properties of Nonparametric Tests for Comparing Survival Distributions. Biometrika, 68, 316-319.

## See Also

gsDesign package overview, plot.gsDesign, gsDesign, gsHR

## Examples

```
library(ggplot2)
# consider a trial with
# 2 year maximum follow-up
# 6 month uniform enrollment
# Treatment/placebo hazards = 0.1/0.2 per 1 person-year
# drop out hazard 0.1 per 1 person-year
# alpha = 0.025 (1-sided)
# power = 0.9 (default beta=.1)
ss <- nSurvival(
    lambda1 = . 2, lambda2 = . 1, eta = . 1, Ts = 2, Tr = . 5,
    sided = 1, alpha = .025
)
# group sequential translation with default bounds
# note that delta1 is log hazard ratio; used later in gsBoundSummary summary
x <- gsDesign(
    k = 5, test.type = 2, n.fix = ss$nEvents, nFixSurv = ss$n,
    delta1 = log(ss$lambda2 / ss$lambda1)
)
# boundary plot
plot(x)
# effect size plot
plot(x, plottype = "hr")
# total sample size
x$nSurv
# number of events at analyses
x$n.I
# print the design
x
# overall design summary
cat(summary(x))
# tabular summary of bounds
gsBoundSummary(x, deltaname = "HR", Nname = "Events", logdelta = TRUE)
# approximate number of events required using Schoenfeld's method
# for 2 different hazard ratios
nEvents(hr = c(.5, . 6), tbl = TRUE)
# vector output
nEvents(hr = c(.5, .6))
# approximate power using Schoenfeld's method
# given 2 sample sizes and hr=.6
nEvents(hr = .6, n = c(50, 100), tbl = TRUE)
# vector output
nEvents(hr = . 6, n = c(50, 100))
# approximate hazard ratio corresponding to 100 events and z-statistic of 2
```

```
zn2hr(n = 100, z = 2)
# same when hr0 is 1.1
zn2hr(n = 100, z = 2, hr0 = 1.1)
# same when hr0 is . }9\mathrm{ and hr1 is greater than hr0
zn2hr(n = 100, z = 2, hr0 = .9, hr1 = 1)
# approximate number of events corresponding to z-statistic of 2 and
# estimated hazard ratio of . 5 (or 2)
hrz2n(hr = .5, z = 2)
hrz2n(hr = 2, z = 2)
# approximate z statistic corresponding to 75 events
# and estimated hazard ratio of . 6 (or 1/.6)
# assuming 2-to-1 randomization of experimental to control
hrn2z(hr = .6, n = 75, ratio = 2)
hrn2z(hr = 1/ . 6, n = 75, ratio = 2)
```


## Description

sequentialPValue computes a sequential p-value for a group sequential design using a spending function as described in Maurer and Bretz (2013) and previously defined by Liu and Anderson (2008). It is the minimum of repeated p-values computed at each analysis (Jennison and Turnbull, 2000). This is particularly useful for multiplicity methods such as the graphical method for group sequential designs where sequential p-values for multiple hypotheses can be used as nominal pvalues to plug into a multiplicity graph. A sequential $p$-value is described as the minimum alpha level at which a one-sided group sequential bound would be rejected given interim and final observed results. It is meaningful for both one-sided designs and designs with non-binding futility bounds (test.type 1, 4, 6), but not for 2-sided designs with binding futility bounds (test.type 2,3 or 5 ). Mild restrictions are required on spending functions used, but these are satisfied for commonly used spending functions such as the Lan-DeMets spending function approximating an O'Brien-Fleming bound or a Hwang-Shih-DeCani spending function; see Maurer and Bretz (2013).

## Usage

```
sequentialPValue(
    gsD = gsDesign(),
    n.I = NULL,
    Z = NULL,
    usTime = NULL,
    interval = c(1e-05, 0.9999)
    )
```


## Arguments

gsD
Group sequential design generated by gsDesign or gsSurv.

| n.I | Event counts (for time-to-event outcomes) or sample size (for most other de- <br> signs); numeric vector with increasing, positive values with at most one value <br> greater than or equal to largest value in gsD\$n. I; NOTE: if NULL, planned n.I <br> will be used (gsD\$n. I). |
| :--- | :--- |
| Z | Z-value tests corresponding to analyses in n.I; positive values indicate a positive <br> finding; must have the same length as n.I. |
| usTime | Spending time for upper bound at specified analyses; specify default: NULL if <br> this is to be based on information fraction; if not NULL, must have the same |
| length as n.I; increasing positive values with at most 1 greater than or equal to |  |

## Details

Solution is found with a search using uniroot. This finds the maximum alpha-level for which an efficacy bound is crossed, completely ignoring any futility bound.

## Value

Sequential p-value (single numeric one-sided p-value between 0 and 1). Note that if the sequential p-value is less than the lower end of the input interval, the lower of interval will be returned. Similarly, if the sequential p-value is greater than the upper end of the input interval, then the upper end of interval is returned.

## Author(s)

Keaven Anderson

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Liu, Qing, and Keaven M. Anderson. "On adaptive extensions of group sequential trials for clinical investigations." Journal of the American Statistical Association 103.484 (2008): 1621-1630.
Maurer, Willi, and Frank Bretz. "Multiple testing in group sequential trials using graphical approaches." Statistics in Biopharmaceutical Research 5.4 (2013): 311-320.;

## Examples

```
# Derive Group Sequential Design
x <- gsSurv(k = 4, alpha = 0.025, beta = 0.1, timing = c(.5,.65,.8), sfu = sfLDOF,
    sfl = sfHSD, sflpar = 2, lambdaC = log(2)/6, hr = 0.6,
    eta = 0.01 , gamma = c(2.5,5,7.5,10), R = c( 2, 2, 2,6 ),
    T = 30, minfup = 18)
x$n.I
# Analysis at IA2
```

```
sequentialPValue(gsD=x,n.I=c(100,160),Z=c(1.5, 2))
# Use planned spending instead of information fraction; do final analysis
sequentialPValue(gsD=x,n.I=c(100,160,190, 230),Z=c(1.5,2,2.5,3),usTime=x$timing)
# Check bounds for updated design to verify at least one was crossed
xupdate <- gsDesign(maxn.IPlan=max(x$n.I),n.I=c(100,160,190, 230),usTime=x$timing,
    delta=x$delta,delta1=x$delta1,k=4,alpha=x$alpha,test.type=1,
    sfu=x$upper$sf,sfupar=x$upper$param)
```

sfExponential Exponential Spending Function

## Description

The function sfExponential implements the exponential spending function (Anderson and Clark, 2009). Normally sfExponential will be passed to gsDesign in the parameter sfu for the upper bound or sfl for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence. The calling sequence is useful, however, when the user wishes to plot a spending function as demonstrated below in examples.
An exponential spending function is defined for any positive nu and $0 \leq t \leq 1$ as

$$
f(t ; \alpha, \nu)=\alpha(t)=\alpha^{t^{-\nu}}
$$

A value of nu=0.8 approximates an $\mathrm{O}^{\prime}$ Brien-Fleming spending function well.
The general class of spending functions this family is derived from requires a continuously increasing cumulative distribution function defined for $x>0$ and is defined as

$$
f(t ; \alpha, \nu)=1-F\left(F^{-1}(1-\alpha) / t^{\nu}\right) .
$$

The exponential spending function can be derived by letting $F(x)=1-\exp (-x)$, the exponential cumulative distribution function. This function was derived as a generalization of the Lan-DeMets (1983) spending function used to approximate an O'Brien-Fleming spending function (sfLDOF ()),

$$
f(t ; \alpha)=2-2 \Phi\left(\Phi^{-1}(1-\alpha / 2) / t^{1 / 2}\right)
$$

## Usage

sfExponential(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha $=0.025$ for one-sided Type I error specification or alpha=0.1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.
t
A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param A single positive value specifying the nu parameter for which the exponential spending is to be computed; allowable range is $(0,1.5]$.

## Value

An object of type spendfn.

## Note

The manual shows how to use sfExponential() to closely approximate an O'Brien-Fleming design. An example is given below. The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.com>

## References

Anderson KM and Clark JB (2009), Fitting spending functions. Statistics in Medicine; 29:321-327.
Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Lan, KKG and DeMets, DL (1983), Discrete sequential boundaries for clinical trials. Biometrika; 70:659-663.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# use 'best' exponential approximation for k=6 to O'Brien-Fleming design
# (see manual for details)
gsDesign(
    k = 6, sfu = sfExponential, sfupar = 0.7849295,
    test.type = 2
)$upper$bound
# show actual O'Brien-Fleming bound
gsDesign(k = 6, sfu = "OF", test.type = 2)$upper$bound
# show Lan-DeMets approximation
# (not as close as sfExponential approximation)
gsDesign(k = 6, sfu = sfLDOF, test.type = 2)$upper$bound
# plot exponential spending function across a range of values of interest
t <- 0:100 / 100
plot(t, sfExponential(0.025, t, 0.8)$spend,
    xlab = "Proportion of final sample size",
    ylab = "Cumulative Type I error spending",
    main = "Exponential Spending Function Example", type = "l"
)
lines(t, sfExponential(0.025, t, 0.5)$spend, lty = 2)
```

```
lines(t, sfExponential(0.025, t, 0.3)$spend, lty = 3)
lines(t, sfExponential(0.025, t, 0.2)$spend, lty = 4)
lines(t, sfExponential(0.025, t, 0.15)$spend, lty = 5)
legend(
    x = c(.0, . 3), y = .025 * c(.7, 1), lty = 1:5,
    legend = c(
            "nu = 0.8", "nu = 0.5", "nu = 0.3", "nu = 0.2",
            "nu = 0.15"
        )
)
text(x = .59, y = .95 * .025, labels = "<--approximates O'Brien-Fleming")
```


## Description

The function sfHSD implements a Hwang-Shih-DeCani spending function. This is the default spending function for gsDesign(). Normally it will be passed to gsDesign in the parameter sfu for the upper bound or sfl for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence. The calling sequence is useful, however, when the user wishes to plot a spending function as demonstrated below in examples.
A Hwang-Shih-DeCani spending function takes the form

$$
f(t ; \alpha, \gamma)=\alpha\left(1-e^{-\gamma t}\right) /\left(1-e^{-\gamma}\right)
$$

where $\gamma$ is the value passed in param. A value of $\gamma=-4$ is used to approximate an O'BrienFleming design (see sfExponential for a better fit), while a value of $\gamma=1$ approximates a Pocock design well.

## Usage

sfHSD(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha $=0.025$ for one-sided Type I error specification or alpha=0.1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.
$t \quad$ A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param A single real value specifying the gamma parameter for which Hwang-ShihDeCani spending is to be computed; allowable range is [-40, 40]

## Value

An object of type spendfn. See Spending_Function_Overview for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.com>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# design a 4-analysis trial using a Hwang-Shih-DeCani spending function
# for both lower and upper bounds
x <- gsDesign(k = 4, sfu = sfHSD, sfupar = -2, sfl = sfHSD, sflpar = 1)
# print the design
x
# since sfHSD is the default for both sfu and sfl,
# this could have been written as
x <- gsDesign(k = 4, sfupar = -2, sflpar = 1)
# print again
x
# plot the spending function using many points to obtain a smooth curve
# show default values of gamma to see how the spending function changes
# also show gamma=1 which is supposed to approximate a Pocock design
t <- 0:100 / 100
plot(t, sfHSD(0.025, t, -4)$spend,
    xlab = "Proportion of final sample size",
    ylab = "Cumulative Type I error spending",
    main = "Hwang-Shih-DeCani Spending Function Example", type = "l"
)
lines(t, sfHSD(0.025, t, -2)$spend, lty = 2)
lines(t, sfHSD(0.025, t, 1)$spend, lty = 3)
legend(
    x = c(.0, . 375), y = .025 * c(.8, 1), lty = 1:3,
    legend = c("gamma= -4", "gamma= -2", "gamma= 1")
)
```


## Description

Lan and DeMets (1983) first published the method of using spending functions to set boundaries for group sequential trials. In this publication they proposed two specific spending functions: one to approximate an O'Brien-Fleming design and the other to approximate a Pocock design. The spending function to approximate $\mathrm{O}^{\prime}$ Brien-Fleming has been generalized as proposed by Liu, et al (2012)

With param=1=rho, the Lan-DeMets (1983) spending function to approximate an O'Brien-Fleming bound is implemented in the function ( $\operatorname{sfLDOF}()$ ):

$$
f(t ; \alpha)=2-2 \Phi\left(\Phi^{-1}(1-\alpha / 2) / t^{\rho / 2}\right)
$$

For rho otherwise in [.005, 2], this is the generalized version of Liu et al (2012). For param outside of $[.005,2]$, rho is set to 1 . The Lan-DeMets (1983) spending function to approximate a Pocock design is implemented in the function sfLDPocock ():

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

As shown in examples below, other spending functions can be used to ge $t$ as good or better approximations to Pocock and O'Brien-Fleming bounds. In particular, O'Brien-Fleming bounds can be closely approximated using sfExponential.

## Usage

sfLDOF (alpha, t, param = NULL)
sfLDPocock(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha= 0.025 for one-sided Type I error specification or alpha=0.1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.
t
A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param This parameter is not used for sfLDPocock, not required for sfLDOF and need not be specified. For sfLDPocock it is here so that the calling sequence conforms to the standard for spending functions used with gsDesign(). For sfLDOF it will default to 1 (Lan-DeMets function to approximate O'Brien-Fleming) if NULL or if outside of the range $[.005,2]$. otherwise, it will be use to set rho from Liu et al (2012).

## Value

An object of type spendfn. See spending functions for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.
Lan, KKG and DeMets, DL (1983), Discrete sequential boundaries for clinical trials. Biometrika;70: 659-663.
Liu, Q, Lim, P, Nuamah, I, and Li, Y (2012), On adaptive error spending approach for group sequential trials with random information levels. Journal of biopharmaceutical statistics; 22(4), 687-699.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# 2-sided, symmetric 6-analysis trial Pocock
# spending function approximation
gsDesign(k = 6, sfu = sfLDPocock, test.type = 2)$upper$bound
# show actual Pocock design
gsDesign(k = 6, sfu = "Pocock", test.type = 2)$upper$bound
# approximate Pocock again using a standard
# Hwang-Shih-DeCani approximation
gsDesign(k = 6, sfu = sfHSD, sfupar = 1, test.type = 2)$upper$bound
# use 'best' Hwang-Shih-DeCani approximation for Pocock, k=6;
# see manual for details
gsDesign(k = 6, sfu = sfHSD, sfupar = 1.3354376, test.type = 2)$upper$bound
# 2-sided, symmetric 6-analysis trial
# O'Brien-Fleming spending function approximation
gsDesign(k = 6, sfu = sfLDOF, test.type = 2)$upper$bound
# show actual O'Brien-Fleming bound
gsDesign(k = 6, sfu = "OF", test.type = 2)$upper$bound
```

```
# approximate again using a standard Hwang-Shih-DeCani
# approximation to O'Brien-Fleming
x <- gsDesign(k = 6, test.type = 2)
x$upper$bound
x$upper$param
# use 'best' exponential approximation for k=6; see manual for details
gsDesign(
    k = 6, sfu = sfExponential, sfupar = 0.7849295,
    test.type = 2
)$upper$bound
# plot spending functions for generalized Lan-DeMets approximation of
ti <-(0:100)/100
rho <- c(.05,.5,1,1.5,2,2.5,3:6,8,10,12.5,15,20,30, 200)/10
df <- NULL
for(r in rho){
    df <- rbind(df,data.frame(t=ti,rho=r,alpha=.025, spend=sfLDOF (alpha=.025,t=ti,param=r)$spend))
}
ggplot(df, aes(x=t,y=spend, col=as.factor(rho)))+
    geom_line()+
    guides(col=guide_legend(expression(rho)))+
    ggtitle("Generalized Lan-DeMets O'Brien-Fleming Spending Function")
```

    sfLinear Piecewise Linear and Step Function Spending Functions
    
## Description

The function sfLinear () allows specification of a piecewise linear spending function. The function sfStep() specifies a step function spending function. Both functions provide complete flexibility in setting spending at desired timepoints in a group sequential design. Normally these function will be passed to gsDesign() in the parameter sfu for the upper bound or $s f l$ for the lower bound to specify a spending function family for a design. When passed to gsDesign(), the value of param would be passed to sfLinear() or sfStep() through the gsDesign() arguments sfupar for the upper bound and sflpar for the lower bound.

Note that $\operatorname{sfStep}()$ allows setting a particular level of spending when the timing is not strictly known; an example shows how this can inflate Type I error when timing of analyses are changed based on knowing the treatment effect at an interim.

## Usage

sfLinear(alpha, t, param)
sfStep(alpha, t, param)

## Arguments

$$
\begin{array}{ll}
\text { alpha } & \begin{array}{l}
\text { Real value }>0 \text { and no more than } 1 \text {. Normally, alpha }=0.025 \text { for one-sided Type } \\
\text { I error specification or alpha= } 0.1 \text { for Type II error specification. However, this } \\
\text { could be set to } 1 \text { if for descriptive purposes you wish to see the proportion of } \\
\text { spending as a function of the proportion of sample size or information. }
\end{array} \\
\text { param } & \begin{array}{l}
\text { A vector of points with increasing values from } 0 \text { to } 1 \text {, inclusive. Values of the } \\
\text { proportion of sample size or information for which the spending function will } \\
\text { be computed. }
\end{array} \\
\begin{array}{l}
\text { A vector with a positive, even length. Values must range from } 0 \text { to } 1 \text {, inclu- } \\
\text { sive. Letting } m<-l e n g t h(p a r a m / 2), ~ t h e ~ f i r s t ~
\end{array} \mathrm{~m} \text { points in param specify increas- } \\
\text { ing values strictly between } 0 \text { and } 1 \text { corresponding to interim timing (propor- } \\
\text { tion of final total statistical information). The last } m \text { points in param specify } \\
\text { non-decreasing values from } 0 \text { to } 1 \text {, inclusive, with the cumulative proportion of } \\
\text { spending at the specified timepoints. }
\end{array}
$$

## Value

An object of type spendfn. The cumulative spending returned in sfLinear\$spend is 0 for $t<=$ 0 and alpha for $t>=1$. For $t$ between specified points, linear interpolation is used to determine sfLinear\$spend.
The cumulative spending returned in sfStep $\$$ spend is 0 for $t<p a r a m[1]$ and alpha for $t>=1$. Letting $m<-$ length (param/2), for $i=1,2, \ldots m-1$ and param[i]<= $t<\operatorname{param}[i+1]$, the cumulative spending is set at alpha * param[i+m] (also for param[m]<=t<1).

Note that if param[2m] is 1 , then the first time an analysis is performed after the last proportion of final planned information (param[m]) will be the final analysis, using any remaining error that was not previously spent.

See Spending_Function_Overview for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# set up alpha spending and beta spending to be piecewise linear
sfupar <- c(.2, .4, .05, .2)
sflpar <- c(.3, .5, .65, .5, .75, .9)
x <- gsDesign(sfu = sfLinear, sfl = sfLinear, sfupar = sfupar, sflpar = sflpar)
plot(x, plottype = "sf")
x
# now do an example where there is no lower-spending at interim 1
# and no upper spending at interim 2
sflpar <- c(1 / 3, 2 / 3, 0, .25)
sfupar <- c(1 / 3, 2 / 3, .1, .1)
x <- gsDesign(sfu = sfLinear, sfl = sfLinear, sfupar = sfupar, sflpar = sflpar)
plot(x, plottype = "sf")
X
# now do an example where timing of interims changes slightly, but error spending does not
# also, spend all alpha when at least >=90 percent of final information is in the analysis
sfupar <- c(.2, .4, .9, ((1:3) / 3)^3)
x <- gsDesign(k = 3, n.fix = 100, sfu = sfStep, sfupar = sfupar, test.type = 1)
plot(x, pl = "sf")
# original planned sample sizes
ceiling(x$n.I)
# cumulative spending planned at original interims
cumsum(x$upper$spend)
# change timing of analyses;
# note that cumulative spending "P(Cross) if delta=0" does not change from cumsum(x$upper$spend)
# while full alpha is spent, power is reduced by reduced sample size
y <- gsDesign(
    k = 3, sfu = sfStep, sfupar = sfupar, test.type = 1,
    maxn.IPlan = x$n.I[x$k], n.I = c(30, 70, 95),
    n.fix = x$n.fix
)
# note that full alpha is used, but power is reduced due to lowered sample size
gsBoundSummary(y)
# now show how step function can be abused by 'adapting' stage 2 sample size based on interim result
x<- gsDesign(k = 2, delta = .05, sfu = sfStep, sfupar = c(.02, .001), timing = .02, test.type = 1)
# spending jumps from miniscule to full alpha at first analysis after interim 1
plot(x, pl = "sf")
# sample sizes at analyses:
ceiling(x$n.I)
# simulate 1 million stage 1 sum of 178 Normal(0,1) random variables
# Normal(0,Variance=178) under null hypothesis
s1 <- rnorm(1000000, 0, sqrt(178))
# compute corresponding z-values
z1 <- s1 / sqrt(178)
# set stage 2 sample size to 1 if z1 is over final bound, otherwise full sample size
n2 <- rep(1, 1000000)
n2[z1 < 1.96] <- ceiling(x$n.I[2]) - ceiling(178)
# now sample n2 observations for second stage
```

```
s2 <- rnorm(1000000, 0, sqrt(n2))
# add sum and divide by standard deviation
z2 <- (s1 + s2) / (sqrt(178 + n2))
# By allowing full spending when final analysis is either
# early or late depending on observed interim z1,
# Type I error is now almost twice the planned . }02
sum(z1 >= x$upper$bound[1] | z2 >= x$upper$bound[2]) / 1000000
# if stage 2 sample size is random and independent of z1 with same frequency,
# this is not a problem
s1alt <- rnorm(1000000, 0, sqrt(178))
z1alt <- s1alt / sqrt(178)
z2alt <- (s1alt + s2) / sqrt(178 + n2)
sum(z1alt >= x$upper$bound[1] | z2alt >= x$upper$bound[2]) / 1000000
```


## Description

The functions sfLogistic(), sfNormal(), sfExtremeValue(), sfExtremeValue2(), sfCauchy(), and sfBetaDist() are all 2-parameter spending function families. These provide increased flexibility in some situations where the flexibility of a one-parameter spending function family is not sufficient. These functions all allow fitting of two points on a cumulative spending function curve; in this case, four parameters are specified indicating an x and a y coordinate for each of 2 points. Normally each of these functions will be passed to gsDesign() in the parameter sfu for the upper bound or sfl for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence. The calling sequence is useful, however, when the user wishes to plot a spending function as demonstrated in the examples; note, however, that an automatic $\alpha$ - and $\beta$-spending function plot is also available.
sfBetaDist(alpha, $t$, param) is simply alpha times the incomplete beta cumulative distribution function with parameters $a$ and $b$ passed in param evaluated at values passed in $t$.
The other spending functions take the form

$$
f(t ; \alpha, a, b)=\alpha F\left(a+b F^{-1}(t)\right)
$$

where $F()$ is a cumulative distribution function with values $>0$ on the real line (logistic for sfLogistic(), normal for sfNormal(), extreme value for sfExtremeValue() and Cauchy for sfCauchy ()) and $F^{-1}()$ is its inverse.
For the logistic spending function this simplifies to

$$
f(t ; \alpha, a, b)=\alpha\left(1-\left(1+e^{a}(t /(1-t))^{b}\right)^{-1}\right)
$$

For the extreme value distribution with

$$
F(x)=\exp (-\exp (-x))
$$

this simplifies to

$$
f(t ; \alpha, a, b)=\alpha \exp \left(-e^{a}(-\ln t)^{b}\right)
$$

Since the extreme value distribution is not symmetric, there is also a version where the standard distribution is flipped about 0 . This is reflected in sfExtremeValue2() where

$$
F(x)=1-\exp (-\exp (x))
$$

## Usage

```
sfLogistic(alpha, t, param)
    sfBetaDist(alpha, t, param)
    sfCauchy(alpha, t, param)
    sfExtremeValue(alpha, t, param)
    sfExtremeValue2(alpha, t, param)
    sfNormal(alpha, t, param)
```


## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha=0.025 for one-sided Type I error specification or alpha=0.1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size or information.
t
A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size or information for which the spending function will be computed.
param In the two-parameter specification, sfBetaDist() requires 2 positive values, while sfLogistic(), sfNormal(), sfExtremeValue(),
sfExtremeValue2() and sfCauchy () require the first parameter to be any real value and the second to be a positive value. The four parameter specification is $\mathrm{c}(\mathrm{t} 1, \mathrm{t} 2, \mathrm{u} 1, \mathrm{u} 2)$ where the objective is that $\mathrm{sf}(\mathrm{t} 1)=\mathrm{alpha*u} 1$ and $\mathrm{sf}(\mathrm{t} 2)=a l \mathrm{pha*u}$. In this parameterization, all four values must be between 0 and 1 and $t 1<t 2$, u1 < u2.

## Value

An object of type spendfn. See Spending_Function_Overview for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven \_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

gsDesign

## Examples

```
library(ggplot2)
# design a 4-analysis trial using a Kim-DeMets spending function
# for both lower and upper bounds
x <- gsDesign(k = 4, sfu = sfPower, sfupar = 3, sfl = sfPower, sflpar = 1.5)
# print the design
x
# plot the alpha- and beta-spending functions
plot(x, plottype = 5)
# start by showing how to fit two points with sfLogistic
# plot the spending function using many points to obtain a smooth curve
# note that curve fits the points x=.1, y=.01 and x=.4, y=.1
# specified in the 3rd parameter of sfLogistic
t <- 0:100 / 100
plot(t, sfLogistic(1, t, c(.1, .4, .01, .1))$spend,
    xlab = "Proportion of final sample size",
    ylab = "Cumulative Type I error spending",
    main = "Logistic Spending Function Examples",
    type = "l", cex.main = . }
)
lines(t, sfLogistic(1, t, c(.01, .1, .1, .4))$spend, lty = 2)
# now just give a=0 and b=1 as 3rd parameters for sfLogistic
lines(t, sfLogistic(1, t, c(0, 1))$spend, lty = 3)
# try a couple with unconventional shapes again using
# the xy form in the 3rd parameter
lines(t, sfLogistic(1, t, c(.4, .6, .1, .7))$spend, lty = 4)
lines(t, sfLogistic(1, t, c(.1, .7, .4, .6))$spend, lty = 5)
legend(
    x = c(.0, .475), y = c(.76, 1.03), lty = 1:5,
    legend = c(
            "Fit (.1, 01) and (.4, .1)", "Fit (.01, .1) and (.1, .4)",
            "a=0, b=1", "Fit (.4, .1) and (.6, .7)",
            "Fit (.1, .4) and (.7, .6)"
    )
)
# set up a function to plot comparsons of all
# 2-parameter spending functions
```

```
plotsf <- function(alpha, t, param) {
    plot(t, sfCauchy(alpha, t, param)$spend,
        xlab = "Proportion of enrollment",
        ylab = "Cumulative spending", type = "l", lty = 2
    )
    lines(t, sfExtremeValue(alpha, t, param)$spend, lty = 5)
    lines(t, sfLogistic(alpha, t, param)$spend, lty = 1)
    lines(t, sfNormal(alpha, t, param)$spend, lty = 3)
    lines(t, sfExtremeValue2(alpha, t, param)$spend, lty = 6, col = 2)
    lines(t, sfBetaDist(alpha, t, param)$spend, lty = 7, col = 3)
    legend(
        x = c(.05, .475), y = .025 * c(.55, .9),
        lty = c(1, 2, 3, 5, 6, 7),
        col = c(1, 1, 1, 1, 2, 3),
        legend = c(
            "Logistic", "Cauchy", "Normal", "Extreme value",
            "Extreme value 2", "Beta distribution"
        )
    )
}
# do comparison for a design with conservative early spending
# note that Cauchy spending function is quite different
# from the others
param <- c(.25, .5, .05, .1)
plotsf(.025, t, param)
```

sfPoints

Pointwise Spending Function

## Description

The function sfPoints implements a spending function with values specified for an arbitrary set of specified points. It is now recommended to use sfLinear rather than sfPoints. Normally sfPoints will be passed to gsDesign in the parameter sfu for the upper bound or $s f l$ for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence, just the points they wish to specify. If using sfPoints() in a design, it is recommended to specify how to interpolate between the specified points (e.g,, linear interpolation); also consider fitting smooth spending functions; see Spending_Function_Overview.

## Usage

sfPoints(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha= 0.025 for one-sided Type I error specification or alpha=0. 1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.

A vector of points with increasing values from $>0$ and $<=1$. Values of the proportion of sample size/information for which the spending function will be computed.
param A vector of the same length as $t$ specifying the cumulative proportion of spending to corresponding to each point in $t$; must be $>=0$ and $<=1$.

## Value

An object of type spendfn. See spending functions for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview, sfLogistic

## Examples

```
library(ggplot2)
# example to specify spending on a pointwise basis
x <- gsDesign(
    k = 6, sfu = sfPoints, sfupar = c(.01, .05, .1, . 25, .5, 1),
    test.type = 2
)
x
# get proportion of upper spending under null hypothesis
# at each analysis
y <- x$upper$prob[, 1] / . }02
# change to cumulative proportion of spending
for (i in 2:length(y))
    y[i] <- y[i - 1] + y[i]
# this should correspond to input sfupar
round(y, 6)
# plot these cumulative spending points
plot(1:6 / 6, y,
```

```
    main = "Pointwise spending function example",
    xlab = "Proportion of final sample size",
    ylab = "Cumulative proportion of spending",
    type = "p"
)
# approximate this with a t-distribution spending function
# by fitting 3 points
tx <- 0:100 / 100
lines(tx, sfTDist(1, tx, c(c(1, 3, 5) / 6, .01, .1, .5))$spend)
text(x = .6, y = .9, labels = "Pointwise Spending Approximated by")
text(x = .6, y = .83, "t-Distribution Spending with 3-point interpolation")
# example without lower spending at initial interim or
# upper spending at last interim
x <- gsDesign(
    k = 3, sfu = sfPoints, sfupar = c(.25, .25),
    sfl = sfPoints, sflpar = c(0, .25)
)
X
```

```
sfPower
```


## Description

The function sfPower() implements a Kim-DeMets (power) spending function. This is a flexible, one-parameter spending function recommended by Jennison and Turnbull (2000). Normally it will be passed to gsDesign() in the parameter sfu for the upper bound or $s f l$ for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence. The calling sequence is useful, however, when the user wishes to plot a spending function as demonstrated below in examples.
A Kim-DeMets spending function takes the form

$$
f(t ; \alpha, \rho)=\alpha t^{\rho}
$$

where $\rho$ is the value passed in param. See examples below for a range of values of $\rho$ that may be of interest (param=0.75 to 3 are documented there).

## Usage

sfPower(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha= 0.025 for one-sided Type I error specification or alpha=0. 1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.
t
A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param A single, positive value specifying the $\rho$ parameter for which Kim-DeMets spending is to be computed; allowable range is $(0,15$ ]

## Value

An object of type spendfn. See Spending_Function_Overview for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# design a 4-analysis trial using a Kim-DeMets spending function
# for both lower and upper bounds
x <- gsDesign(k = 4, sfu = sfPower, sfupar = 3, sfl = sfPower, sflpar = 1.5)
# print the design
x
# plot the spending function using many points to obtain a smooth curve
# show rho=3 for approximation to 0'Brien-Fleming and rho=. }75\mathrm{ for
# approximation to Pocock design.
# Also show rho=2 for an intermediate spending.
# Compare these to Hwang-Shih-DeCani spending with gamma=-4, -2, 1
t <- 0:100 / 100
plot(t, sfPower(0.025, t, 3)$spend,
    xlab = "Proportion of sample size",
    ylab = "Cumulative Type I error spending",
    main = "Kim-DeMets (rho) versus Hwang-Shih-DeCani (gamma) Spending",
    type = "l", cex.main = . }
)
lines(t, sfPower(0.025, t, 2)$spend, lty = 2)
```

```
lines(t, sfPower(0.025, t, 0.75)$spend, lty = 3)
lines(t, sfHSD(0.025, t, 1)$spend, lty = 3, col = 2)
lines(t, sfHSD(0.025, t, -2)$spend, lty = 2, col = 2)
lines(t, sfHSD(0.025, t, -4)$spend, lty = 1, col = 2)
legend(
    x = c(.0, .375), y = .025 * c(.65, 1), lty = 1:3,
    legend = c("rho= 3", "rho= 2", "rho= 0.75")
)
legend(
    x = c(.0, .357), y = .025 * c(.65, .85), lty = 1:3, bty = "n", col = 2,
    legend = c("gamma= -4", "gamma= -2", "gamma=1")
)
```

sfTDist t-distribution Spending Function

## Description

The function sfTDist() provides perhaps the maximum flexibility among spending functions provided in the gsDesign package. This function allows fitting of three points on a cumulative spending function curve; in this case, six parameters are specified indicating an $x$ and a y coordinate for each of 3 points. Normally this function will be passed to gsDesign() in the parameter sfu for the upper bound or $s f l$ for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence. The calling sequence is useful, however, when the user wishes to plot a spending function as demonstrated below in examples.

The t -distribution spending function takes the form

$$
f(t ; \alpha)=\alpha F\left(a+b F^{-1}(t)\right)
$$

where $F()$ is a cumulative t -distribution function with df degrees of freedom and $F^{-1}()$ is its inverse.

## Usage

sfTDist(alpha, t, param)

## Arguments

alpha $\quad$ Real value $>0$ and no more than 1 . Normally, alpha= 0.025 for one-sided Type I error specification or alpha=0.1 for Type II error specification. However, this could be set to 1 if for descriptive purposes you wish to see the proportion of spending as a function of the proportion of sample size/information.
t
A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param In the three-parameter specification, the first paramater (a) may be any real value, the second (b) any positive value, and the third parameter ( $\mathrm{df}=\mathrm{degrees}$ of freedom) any real value 1 or greater. When gsDesign() is called with a t -distribution spending function, this is the parameterization printed. The five parameter specification is $c(t 1, t 2, u 1, u 2, d f)$ where the objective is that the resulting cumulative proportion of spending at $t$ represented by $s f(t)$ satisfies $s f(t 1)=a l p h a * u 1, s f(t 2)=a l p h a * u 2$. The $t$-distribution used has df degrees of freedom. In this parameterization, all the first four values must be between 0 and 1 and $\mathrm{t} 1<\mathrm{t} 2, \mathrm{u} 1<\mathrm{u} 2$. The final parameter is any real value of 1 or more. This parameterization can fit any two points satisfying these requirements. The six parameter specification attempts to fit 3 points, but does not have flexibility to fit any three points. In this case, the specification for param is $\mathrm{c}(\mathrm{t} 1, \mathrm{t} 2, \mathrm{t} 3, \mathrm{u} 1, \mathrm{u} 2, \mathrm{u} 3)$ where the objective is that $\mathrm{sf}(\mathrm{t} 1)=\mathrm{alpha*u} 1, \mathrm{sf}(\mathrm{t} 2)=\mathrm{alpha*u} 2$, and $s f(t 3)=a l p h a * u 3$. See examples to see what happens when points are specified that cannot be fit.

## Value

An object of type spendfn. See spending functions for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.com>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
library(ggplot2)
# 3-parameter specification: a, b, df
sfTDist(1, 1:5 / 6, c(-1, 1.5, 4))$spend
# 5-parameter specification fits 2 points, in this case
# the 1st 2 interims are at 25% and 50% of observations with
# cumulative error spending of 10% and 20%, respectively
# final parameter is df
sfTDist(1, 1:3 / 4, c(.25, .5, .1, .2, 4))$spend
# 6-parameter specification fits 3 points
```

```
# Interims are at 25%. 50% and 75% of observations
# with cumulative spending of 10%, 20% and 50%, respectively
# Note: not all 3 point combinations can be fit
sfTDist(1, 1:3 / 4, c(.25, .5, . 75, .1, .2, .5))$spend
# Example of error message when the 3-points specified
# in the 6-parameter version cannot be fit
try(sfTDist(1, 1:3 / 4, c(.25, .5, .75, .1, .2, .3))$errmsg)
# sfCauchy (sfTDist with 1 df) and sfNormal (sfTDist with infinite df)
# show the limits of what sfTdist can fit
# for the third point are u3 from 0.344 to 0.6 when t3=0.75
sfNormal(1, 1:3 / 4, c(.25, .5, .1, .2))$spend[3]
sfCauchy(1, 1:3 / 4, c(.25, .5, .1, .2))$spend[3]
# plot a few t-distribution spending functions fitting
# t=0.25, . 5 and u=0.1, 0.2
# to demonstrate the range of flexibility
t <- 0:100 / 100
plot(t, sfTDist(0.025, t, c(.25, .5, .1, .2, 1))$spend,
    xlab = "Proportion of final sample size",
    ylab = "Cumulative Type I error spending",
    main = "t-Distribution Spending Function Examples", type = "l"
)
lines(t, sfTDist(0.025, t, c(.25, .5, .1, .2, 1.5))$spend, lty = 2)
lines(t, sfTDist(0.025, t, c(.25, .5, .1, .2, 3))$spend, lty = 3)
lines(t, sfTDist(0.025, t, c(.25, .5, .1, .2, 10))$spend, lty = 4)
lines(t, sfTDist(0.025, t, c(.25, .5, .1, .2, 100))$spend, lty = 5)
legend(
    x = c(.0, . 3), y = .025 * c(.7, 1), lty = 1:5,
    legend = c("df = 1", "df = 1.5", "df = 3", "df = 10", "df = 100")
)
```


## Description

The functions sfTruncated() and sfTrimmed apply any other spending function over a restricted range. This allows eliminating spending for early interim analyses when you desire not to stop for the bound being specified; this is usually applied to eliminate early tests for a positive efficacy finding. The truncation can come late in the trial if you desire to stop a trial any time after, say, 90 percent of information is available and an analysis is performed. This allows full Type I error spending if the final analysis occurs early. Both functions set cumulative spending to 0 below a 'spending interval' in the interval [ 0,1 ], and set cumulative spending to 1 above this range. sfTrimmed() otherwise does not change an input spending function that is specified; probably the preferred and more intuitive method in most cases. sfTruncated() resets the time scale on which the input spending function is computed to the 'spending interval.'
sfGapped() allows elimination of analyses after some time point in the trial; see details and examples.
sfTrimmed simply computes the value of the input spending function and parameters in the subrange of $[0,1]$, sets spending to 0 below this range and sets spending to 1 above this range.
sfGapped spends outside of the range provided in trange. Below trange, the input spending function is used. Above trange, full spending is used; i.e., the first analysis performed above the interval in trange is the final analysis. As long as the input spending function is strictly increasing, this means that the first interim in the interval trange is the final interim analysis for the bound being specified.
sfTruncated compresses spending into a sub-range of $[0,1]$. The parameter param\$trange specifies the range over which spending is to occur. Within this range, spending is spent according to the spending function specified in param\$sf along with the corresponding spending function parameter(s) in param\$param. See example using sfLinear that spends uniformly over specified range.

## Usage

sfTruncated(alpha, t, param)
sfTrimmed(alpha, t, param)
sfGapped(alpha, t, param)

## Arguments

$$
\begin{array}{ll}
\text { alpha } & \begin{array}{l}
\text { Real value }>0 \text { and no more than } 1 \text {. Normally, alpha= } 0.025 \text { for one-sided Type } \\
\text { I error specification or alpha=0. } 1 \text { for Type II error specification. However, this } \\
\text { could be set to } 1 \text { if for descriptive purposes you wish to see the proportion of } \\
\text { spending as a function of the proportion of sample size or information. }
\end{array} \\
\mathrm{t} & \begin{array}{l}
\text { A vector of points with increasing values from } 0 \text { to } 1 \text {, inclusive. Values of the } \\
\text { proportion of sample size or information for which the spending function will } \\
\text { be computed. }
\end{array} \\
\begin{array}{l}
\text { a list containing the elements sf (a spendfn object such as sfHSD), trange (the } \\
\text { range over which the spending function increases from } 0 \text { to } 1 ; 0<=\text { trange[1]<trange[2] } \\
<=1 ; \text { for sfGapped, trange[1] must be }>0), \text { and param (null for a spending func- } \\
\text { tion with no parameters or a scalar or vector of parameters needed to fully spec- } \\
\text { ify the spending function in sf). }
\end{array}
\end{array}
$$

## Value

An object of type spendfn. See Spending_Function_Overview for further details.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

Spending_Function_Overview, gsDesign, gsDesign package overview

## Examples

```
# Eliminate efficacy spending forany interim at or before 20 percent of information.
# Complete spending at first interim at or after 80 percent of information.
tx <- (0:100) / 100
s <- sfHSD(alpha = .05, t = tx, param = 1)$spend
x <- data.frame(t = tx, Spending = s, sf = "Original spending")
param <- list(trange = c(.2, .8), sf = sfHSD, param = 1)
s <- sfTruncated(alpha = .05, t = tx, param = param)$spend
x <- rbind(x, data.frame(t = tx, Spending = s, sf = "Truncated"))
s <- sfTrimmed(alpha = .05, t = tx, param = param)$spend
x <- rbind(x, data.frame(t = tx, Spending = s, sf = "Trimmed"))
s <- sfGapped(alpha = .05, t = tx, param = param)$spend
x <- rbind(x, data.frame(t = tx, Spending = s, sf = "Gapped"))
ggplot2::ggplot(x, ggplot2::aes(x = t, y = Spending, col = sf)) +
ggplot2::geom_line()
# now apply the sfTrimmed version in gsDesign
# initially, eliminate the early efficacy analysis
# note: final spend must occur at > next to last interim
x <- gsDesign(
    k = 4, n.fix = 100, sfu = sfTrimmed,
    sfupar = list(sf = sfHSD, param = 1, trange = c(.3, .9))
)
# first upper bound=20 means no testing there
gsBoundSummary(x)
# now, do not eliminate early efficacy analysis
param <- list(sf = sfHSD, param = 1, trange = c(0, .9))
x <- gsDesign(k = 4, n.fix = 100, sfu = sfTrimmed, sfupar = param)
# The above means if final analysis is done a little early, all spending can occur
# Suppose we set calendar date for final analysis based on
# estimated full information, but come up with only 97 pct of plan
xA <- gsDesign(
    k = x$k, n.fix = 100, n.I = c(x$n.I[1:3], .97 * x$n.I[4]),
    test.type = x$test.type,
    maxn.IPlan = x$n.I[x$k],
    sfu = sfTrimmed, sfupar = param
)
```

```
# now accelerate without the trimmed spending function
xNT <- gsDesign(
    k = x$k, n.fix = 100, n.I = c(x$n.I[1:3], .97* x$n.I[4]),
    test.type = x$test.type,
    maxn.IPlan = x$n.I[x$k],
    sfu = sfHSD, sfupar = 1
)
# Check last bound if analysis done at early time
x$upper$bound[4]
# Now look at last bound if done at early time with trimmed spending function
# that allows capture of full alpha
xA$upper$bound[4]
# With original spending function, we don't get full alpha and therefore have
# unnecessarily stringent bound at final analysis
xNT$upper$bound[4]
# note that if the last analysis is LATE, all 3 approaches should give the same
# final bound that has a little larger z-value
xlate <- gsDesign(
    k = x$k, n.fix = 100, n.I = c(x$n.I[1:3], 1.25* x$n.I[4]),
    test.type = x$test.type,
    maxn.IPlan = x$n.I[x$k],
    sfu = sfHSD, sfupar = 1
)
xlate$upper$bound[4]
# eliminate futility after the first interim analysis
# note that by setting trange[1] to .2, the spend at t=.2 is used for the first
# interim at or after 20 percent of information
x<- gsDesign(n.fix = 100, sfl = sfGapped, sflpar = list(trange = c(.2, .9), sf = sfHSD, param = 1))
```

Spending_Function_Overview
4.0: Spending function overview

## Description

Spending functions are used to set boundaries for group sequential designs. Using the spending function approach to design offers a natural way to provide interim testing boundaries when unplanned interim analyses are added or when the timing of an interim analysis changes. Many standard and investigational spending functions are provided in the gsDesign package. These offer a great deal of flexibility in setting up stopping boundaries for a design.

## Arguments

alpha Real value $>0$ and no more than 1. Defaults in calls to gsDesign() are alpha=0.025 for one-sided Type I error specification and alpha=0.1 for Type II error specification. However, this could be set to 1 if , for descriptive purposes, you wish to see the proportion of spending as a function of the proportion of sample size/information.
$t \quad$ A vector of points with increasing values from 0 to 1 , inclusive. Values of the proportion of sample size/information for which the spending function will be computed.
param A single real value or a vector of real values specifying the spending function parameter(s); this must be appropriately matched to the spending function specified.
object A spendfn object to be summarized.
... Not currently used.

## Details

The summary () function for spendfn objects provides a brief textual summary of a spending function or boundary used for a design.

Spending functions have three arguments as noted above and return an object of type spendfn. Normally a spending function will be passed to gsDesign() in the parameter sfu for the upper bound and $s f l$ for the lower bound to specify a spending function family for a design. In this case, the user does not need to know the calling sequence - only how to specify the parameter(s) for the spending function. The calling sequence is useful when the user wishes to plot a spending function as demonstrated below in examples. In addition to using supplied spending functions, a user can write code for a spending function. See examples.

## Value

spendingFunction and spending functions in general produce an object of type spendfn.
name A character string with the name of the spending function.
param any parameters used for the spending function.
parname a character string or strings with the name(s) of the parameter(s) in param.
sf the spending function specified.
spend a vector of cumulative spending values corresponding to the input values in $t$.
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 probabilities of boundary crossing at $i$-th analysis for $j$-th theta value input to gsDesign() in prob[i,j].

Note
The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven \_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

gsDesign, sfHSD, sfPower, sfLogistic, sfExponential, sfTruncated, gsDesign package overview

## Examples

```
library(ggplot2)
# Example 1: simple example showing what mose users need to know
# design a 4-analysis trial using a Hwang-Shih-DeCani spending function
# for both lower and upper bounds
x <- gsDesign(k=4, sfu=sfHSD, sfupar=-2, sfl=sfHSD, sflpar=1)
# print the design
X
# summarize the spending functions
summary (x$upper)
summary(x$lower)
# plot the alpha- and beta-spending functions
plot(x, plottype=5)
# what happens to summary if we used a boundary function design
x <- gsDesign(test.type=2, sfu="OF")
y <- gsDesign(test.type=1, sfu="WT", sfupar=.25)
summary(x$upper)
summary(y$upper)
# Example 2: advanced example: writing a new spending function
# Most users may ignore this!
# 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)
{
    # check inputs
    checkVector(param, "numeric", c(0, Inf), c(FALSE, TRUE))
    if (length(param) !=2) stop(
    "b-dist example spending function parameter must be of length 2")
    # set spending using cumulative beta distribution and return
    x <- list(name="B-dist example", param=param, parname=c("a", "b"),
            sf=sfbdist, spend=alpha *
                pbeta(t, param[1], param[2]), bound=NULL, prob=NULL)
    class(x) <- "spendfn"
```

```
    x
}
# now try it out!
# plot some example beta (lower bound) spending functions using
# the beta distribution spending function
t <- 0:100/100
plot(t, sfbdist(1, t, c(2, 1))$spend, type="l",
    xlab="Proportion of information",
    ylab="Cumulative proportion of total spending",
    main="Beta distribution Spending Function Example")
lines(t, sfbdist(1, t, c(6, 4))$spend, lty=2)
lines(t, sfbdist(1, t, c(.5, .5))$spend, lty=3)
lines(t, sfbdist(1, t, c(.6, 2))$spend, lty=4)
legend(x=c(.65, 1), y=1 * c(0, .25), lty=1:4,
    legend=c("a=2, b=1","a=6, b=4","a=0.5, b=0.5","a=0.6, b=2"))
```

summary.gsDesign Bound Summary and Z-transformations

## Description

A tabular summary of a group sequential design's bounds and their properties are often useful. The 'vintage' print.gsDesign() function provides a complete but minimally formatted summary of a group sequential design derived by gsDesign(). A brief description of the overall design can also be useful (summary.gsDesign(). A tabular summary of boundary characteristics oriented only towards LaTeX output is produced by xtable.gsSurv. More flexibility is provided by gsBoundSummary () which produces a tabular summary of a user-specifiable set of packageprovided boundary properties in a data frame. This can also be used to along with functions such as print.data.frame(), write.table(), write.csv(), write.csv2() or, from the RTF package, addTable.RTF () (from the rtf package) to produce console or R Markdown output or output to a variety of file types. xprint() is provided for LaTeX output by setting default options for print.xtable() when producing tables summarizing design bounds.

Individual transformation of z-value test statistics for interim and final analyses are obtained from gsBValue(), gsDelta(), gsHR() and gsCPz() for B-values, approximate treatment effect (see details), approximate hazard ratio and conditional power, respectively.
The print.gsDesign function is intended to provide an easier output to review than is available from a simple list of all the output components. The gsBoundSummary function is intended to provide a summary of boundary characteristics that is often useful for evaluating boundary selection; this outputs an extension of the data. frame class that sets up default printing without row names using print.gsBoundSummary. summary.gsDesign, on the other hand, provides a summary of the overall design at a higher level; this provides characteristics not included in the gsBoundSummary summary and no detail concerning interim analysis bounds.
In brief, the computed descriptions of group sequential design bounds are as follows: Z: Standardized normal test statistic at design bound.
p (1-sided): 1-sided p-value for $Z$. This will be computed as the probability of a greater EXCEPT for lower bound when a 2 -sided design is being summarized.
delta at bound: Approximate value of the natural parameter at the bound. The approximate standardized effect size at the bound is generally computed as $Z / \operatorname{sqrt}(n)$. Calling this theta, this is translated to the delta using the values delta 0 and delta 1 from the input x by the formula delta $0+($ delta1-delta0)/theta1*theta where theta1 is the alternate hypothesis value of the standardized parameter. Note that this value will be exponentiated in the case of relative risks, hazard ratios or when the user specifies logdelta=TRUE. In the case of hazard ratios, the value is computed instead by $\operatorname{gsHR}()$ to be consistent with plot.gsDesign(). Similarly, the value is computed by $\operatorname{gsRR}()$ when the relative risk is the natural parameter.
Spending: Incremental error spending at each given analysis. For asymmetric designs, futility bound will have beta-spending summarized. Efficacy bound always has alpha-spending summarized.
$B$-value: $\operatorname{sqr} t(t) * Z$ where $t$ is the proportion of information at the analysis divided by the final analysis planned information. The expected value for $B$-values is directly proportional to $t$.
CP : Conditional power under the estimated treatment difference assuming the interim Z-statistic is at the study bound

CP H1: Conditional power under the alternate hypothesis treatment effect assuming the interim test statistic is at the study bound.
PP: Predictive power assuming the interim test statistic is at the study bound and the input prior distribution for the standardized effect size. This is the conditional power averaged across the posterior distribution for the treatment effect given the interim test statistic value. P\{Cross if delta $=x x\}$ : For each of the parameter values in $x$, the probability of crossing either bound given that treatment effect is computed. This value is cumulative for each bound. For example, the probability of crossing the efficacy bound at or before the analysis of interest.

## Usage

```
## S3 method for class 'gsDesign'
summary(object, information = FALSE, timeunit = "months", ...)
## S3 method for class 'gsDesign'
print(x, ...)
gsBoundSummary(
    x,
    deltaname = NULL,
    logdelta = FALSE,
    Nname = NULL,
    digits = 4,
    ddigits = 2,
    tdigits = 0,
    timename = "Month",
    prior = normalGrid(mu = x$delta/2, sigma = 10/sqrt(x$n.fix)),
    POS = FALSE,
    ratio = NULL,
    exclude = c("B-value", "Spending", "CP", "CP H1", "PP"),
```

```
    r=18,
)
xprint(
    x,
    include.rownames = FALSE,
    hline.after = c(-1, which(x$Value == x[1, ]$Value) - 1, nrow(x)),
    ..
)
## S3 method for class 'gsBoundSummary'
print(x, row.names = FALSE, digits = 4, ...)
gsBValue(z, i, x, ylab = "B-value", ...)
gsDelta(z, i, x, ylab = NULL, ...)
gsRR(z, i, x, ratio = 1, ylab = "Approximate risk ratio", ...)
gsHR(z, i, x, ratio = 1, ylab = "Approximate hazard ratio", ...)
gsCPz(z, i, x, theta = NULL, ylab = NULL, ...)
```


## Arguments

object An item of class gsDesign or gsSurv
information indicator of whether n.I in object represents statistical information rather than sample size or event counts.
timeunit Text string with time units used for time-to-event designs created with gsSurv ()
... This allows many optional arguments that are standard when calling plot for gsBValue, gsDelta, gsHR, gsRR and gsCPz
x
An item of class gsDesign or gsSurv, except for print.gsBoundSummary () where $x$ is an object created by gsBoundSummary () and xprint() which is used with xtable (see examples)
deltaname Natural parameter name. If default NULL is used, routine will default to "HR" when class is gsSurv or if $n$ FixSurv was input when creating $x$ with gsDesign().
logdelta Indicates whether natural parameter is the natural logarithm of the actual parameter. For example, the relative risk or odds-ratio would be put on the logarithmic scale since the asymptotic behavior is 'more normal' than a non-transformed value. As with deltaname, the default will be changed to true if $x$ has class gsDesign or if $n F i x S u r v>0$ was input when $x$ was created by gsDesign(); that is, the natural parameter for a time-to-event endpoint will be on the logarithmic scale.

Nname This will normally be changed to " N " or, if a time-to-event endpoint is used, "Events". Other immediate possibility are "Deaths" or "Information".

| digits | Number of digits past the decimal to be printed in the body of the table. |
| :--- | :--- |
| ddigits | Number of digits past the decimal to be printed for the natural parameter delta. |
| tdigits | Number of digits past the decimal point to be shown for estimated timing of <br> each analysis. |
| timename | Text string indicating time unit. |
| prior | A prior distribution for the standardized effect size. Must be of the format pro- <br> duced by normalGrid(), but can reflect an arbitrary prior distribution. The <br> default reflects a normal prior centered half-way between the null and alternate <br> hypothesis with the variance being equivalent to the treatment effect estimate if 1 <br> percent of the sample size for a fixed design were sampled. The prior is intended |
| to be relatively uninformative. This input will only be applied if PoS=TRUE is in- |  |
| put. |  |$\quad$| This is an indicator of whether or not probability of success (POS) should be |
| :--- |
| estimated at baseline or at each interim based on the prior distribution input in |
| prior. The prior probability of success before the trial starts is the power of the |
| study averaged over the prior distribution for the standardized effect size. The |
| POS after an interim analysis assumes the interim test statistic is an unknown |
| value between the futility and efficacy bounds. Based on this, a posterior distri- |
| bution for the standardized parameter is computed and the conditional power of |
| the trial is averaged over this posterior distribution. |

## Value

gsBValue(), gsDelta(), gsHR() and gsCPz() each returns a vector containing the B-values, approximate treatment effect (see details), approximate hazard ratio and conditional power, respectively, for each value specified by the interim test statistics in $z$ at interim analyses specified in i.
summary returns a text string summarizing the design at a high level. This may be used with gsBoundSummary for a nicely formatted, concise group sequential design description.
gsBoundSummary returns a table in a data frame providing a variety of boundary characteristics. The tabular format makes formatting particularly amenable to place in documents either through direct creation of readable by Word (see the rtf package) or to a csv format readable by spreadsheet software using write.csv.
print.gsDesign prints an overall summary a group sequential design. While the design description is complete, the format is not as 'document friendly' as gsBoundSummary.
print.gsBoundSummary is a simple extension of print. data.frame intended for objects created with gsBoundSummary. The only extension is to make the default to not print row names. This is probably 'not good R style' but may be helpful for many lazy R programmers like the author.

## Note

The manual is not linked to this help file, but is available in library/gsdesign/doc/gsDesignManual.pdf in the directory where R is installed.

## Author(s)

Keaven Anderson <keaven\_anderson@merck.>

## References

Jennison C and Turnbull BW (2000), Group Sequential Methods with Applications to Clinical Trials. Boca Raton: Chapman and Hall.

## See Also

gsDesign, plot.gsDesign, gsProbability, xtable.gsSurv

## Examples

```
library(ggplot2)
# survival endpoint using gsSurv
# generally preferred over nSurv since time computations are shown
xgs <- gsSurv(lambdaC = .2, hr = .5, eta = .1, T = 2, minfup = 1.5)
gsBoundSummary(xgs, timename = "Year", tdigits = 1)
summary(xgs)
# survival endpoint using nSurvival
# NOTE: generally recommend gsSurv above for this!
ss <- nSurvival(
    lambda1 = .2, lambda2 = .1, eta = .1, Ts = 2, Tr = .5,
    sided = 1, alpha = .025, ratio = 2
```

```
)
xs <- gsDesign(nFixSurv = ss$n, n.fix = ss$nEvents, delta1 = log(ss$lambda2 / ss$lambda1))
gsBoundSummary(xs, logdelta = TRUE, ratio = ss$ratio)
# generate some of the above summary statistics for the upper bound
z <- xs$upper$bound
# B-values
gsBValue(z = z, i = 1:3, x = xs)
# hazard ratio
gsHR(z = z, i = 1:3, x = xs)
# conditional power at observed treatment effect
gsCPz(z = z[1:2], i = 1:2, x = xs)
# conditional power at H1 treatment effect
gsCPz(z = z[1:2], i = 1:2, x = xs, theta = xs$delta)
# information-based design
xinfo <- gsDesign(delta = .3, delta1 = .3)
gsBoundSummary(xinfo, Nname = "Information")
# show all available boundary descriptions
gsBoundSummary(xinfo, Nname = "Information", exclude = NULL)
# add intermediate parameter value
xinfo <- gsProbability(d = xinfo, theta = c(0, .15, .3))
class(xinfo) # note this is still as gsDesign class object
gsBoundSummary(xinfo, Nname = "Information")
# now look at a binomial endpoint; specify H0 treatment difference as p1-p2=.05
# now treatment effect at bound (say, thetahat) is transformed to
# xp$delta0 + xp$delta1*(thetahat-xp$delta0)/xp$delta
np <- nBinomial(p1 = .15, p2 = .10)
xp <- gsDesign(n.fix = np, endpoint = "Binomial", delta1 = .05)
summary(xp)
gsBoundSummary(xp, deltaname = "p[C]-p[E]")
# estimate treatment effect at lower bound
# by setting delta0=0 (default) and delta1 above in gsDesign
# treatment effect at bounds is scaled to these differences
# in this case, this is the difference in event rates
gsDelta(z = xp$lower$bound, i = 1:3, xp)
# binomial endpoint with risk ratio estimates
n.fix <- nBinomial(p1 = .3, p2 = .15, scale = "RR")
xrr <- gsDesign(k = 2, n.fix = n.fix, delta1 = log(.15 / .3), endpoint = "Binomial")
gsBoundSummary(xrr, deltaname = "RR", logdelta = TRUE)
gsRR(z = xp$lower$bound, i = 1:3, xrr)
plot(xrr, plottype = "RR")
# delta is odds-ratio: sample size slightly smaller than for relative risk or risk difference
n.fix <- nBinomial(p1 = .3, p2 = .15, scale = "OR")
xOR <- gsDesign(k = 2, n.fix = n.fix, delta1 = log(.15 / . 3/ . 85 * . 7), endpoint = "Binomial")
gsBoundSummary(xOR, deltaname = "OR", logdelta = TRUE)
# for nice LaTeX table output, use xprint
xprint(xtable::xtable(gsBoundSummary(xOR, deltaname = "OR", logdelta = TRUE),
```

    caption = "Table caption."))
    xtable xtable
    
## Description

xtable

Value
an object of class "xtable"

## Index

```
* design
    ciBinomial,5
    condPower, 11
    eEvents,17
    gsBinomialExact, 20
    gsBound, 25
    gsBoundCP, 28
    gsCP,29
    gsDensity, 33
    gsDesign, 36
    gsDesign package overview,43
    gsProbability,44
    nNormal,49
    normalGrid,52
    plot.gsDesign,54
    print.nSurv, 57
    print.nSurvival,66
    sfExponential,73
    sfHSD,75
    sfLDOF,77
    sfLinear,79
    sfLogistic, 82
    sfPoints, }8
    sfPower, }8
    sfTDist,89
    sfTruncated, 91
    Spending_Function_Overview, 94
    summary.gsDesign,97
* programming
    checkLengths, 2
binomialSPRT (gsBinomialExact),20
checkLengths, 2, 3
checkRange, 3, 4
checkRange (checkLengths), 2
checkScalar, 3
checkScalar (checkLengths), 2
checkVector, 3
checkVector (checkLengths), 2
```

ciBinomial, 5
condPower, 11, 29, 32
eEvents, 17
gsBinomialExact, 20
gsBound, 25
gsBound1 (gsBound), 25
gsBoundCP, 28, 32, 35
gsBoundSummary, 42, 61, 64
gsBoundSummary (summary.gsDesign), 97
gsBValue (summary.gsDesign), 97
gsCP, 29, 29
gsCPOS (gsCP), 29
gsCPz (summary.gsDesign), 97
gsDelta (summary.gsDesign), 97
gsDensity, 33
gsDesign, 13, 14, 19, 27, 29, 32, 35, 36,
$44-46,56,63,64,69,74,76,78,80$,
$84,86,88,90,93,96,100,101$
gsDesign package overview, $19,27,42,43$,
$46,51,64,69,74,76,78,80,86,88$,
90, 93, 96
gsHR, 19, 64, 69
gsHR (summary.gsDesign), 97
gsPI (gsCP), 29
gsPOS (gsCP), 29
gsPosterior (gsCP), 29
gsPP, 34
gsPP (gsCP), 29
gsProbability, 24, 27, 29, 32, 35, 42, 44, 44,
56, 101
gsRR (summary.gsDesign), 97
gsSurv (print.nSurv), 57
hGraph, 47
hrn2z (print.nSurvival), 66
hrz2n (print.nSurvival), 66
is.integer, 2
isInteger (checkLengths), 2
nBinomial (ciBinomial), 5
nBinomial1Sample (gsBinomialExact), 20
nEvents (print.nSurvival), 66
nEventsIA (print.nSurv), 57
nNormal, 49
Normal, 9, 42, 64
normalGrid, 30-32, 52
nSurv (print.nSurv), 57
nSurvival, 17, 19, 61, 64
nSurvival (print.nSurvival), 66
plot.binomialSPRT (gsBinomialExact), 20
plot.gsBinomialExact (gsBinomialExact), 20
plot.gsDesign, $19,42,46,54,64,69,101$
plot.gsProbability (plot.gsDesign), 54
plot.ssrCP (condPower), 11
Power.ssrCP (condPower), 11
print.data.frame, 97
print.eEvents (eEvents), 17
print.gsBinomialExact
(gsBinomialExact), 20
print.gsBoundSummary
(summary.gsDesign), 97
print.gsDesign, 36
print.gsDesign (summary.gsDesign), 97
print.gsProbability (gsProbability), 44
print.gsSurv (print.nSurv), 57
print.nSurv, 57
print.nSurvival, 66
print.xtable, 97
sequentialPValue, 71
sfBetaDist (sfLogistic), 82
sfCauchy (sfLogistic), 82
sfExponential, 73, 75, 77, 96
sfExtremeValue (sfLogistic), 82
sfExtremeValue2 (sfLogistic), 82
sfGapped (sfTruncated), 91
sfHSD, 75, 96
sfLDOF, 77
sfLDPocock (sfLDOF), 77
sfLinear, 79
sfLogistic, 82, 86, 96
sfNormal (sfLogistic), 82
sfPoints, 85
sfPower, 87, 96
sfStep (sfLinear), 79
sfTDist, 89
sfTrimmed (sfTruncated), 91
sfTruncated, 91, 96
simBinomial (ciBinomial), 5
Spending_Function_Overview, 38, 41, 42, 60, 74-76, 78, 80, 83, 85, 86, 88, 90, 92, 93, 94
spendingFunction (Spending_Function_Overview), 94
ssrCP, 29, 32
ssrCP (condPower), 11
summary.gsDesign, 97
summary.spendfn
(Spending_Function_Overview), 94
testBinomial (ciBinomial), 5
tEventsIA (print.nSurv), 57
uniroot, 9, 59, 64
varBinomial (ciBinomial), 5
write.csv, 97
write.csv2, 97
write.table, 97
xprint, 61, 64
xprint (summary.gsDesign), 97
xtable, 42, 64, 103
xtable.gsDesign (gsDesign), 36
xtable.gsSurv, 97, 101
xtable.gsSurv (print.nSurv), 57
z2Fisher (condPower), 11
z2NC (condPower), 11
z2Z (condPower), 11
zn2hr (print.nSurvival), 66

