Package ‘PwrGSD’

December 11, 2021

Title Power in a Group Sequential Design
Version 2.3.6
Author Grant Izmirlian <izmirlig@mail.nih.gov>
Depends survival
Description Tools for the evaluation of interim analysis plans for sequentially
monitored trials on a survival endpoint; tools to construct efficacy and
futility boundaries, for deriving power of a sequential design at a specified
alternative, template for evaluating the performance of candidate plans at a
Maintainer Grant Izmirlian <izmirlig@mail.nih.gov>
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2021-12-10 23:40:07 UTC

R topics documented:

agghaz ................................................................. 2
as.boundaries ..................................................... 3
CDFOR2LRR .......................................................... 4
CondPower ........................................................... 4
cpd.PwrGSD .......................................................... 5
CRRtoRR ............................................................. 10
CY2TOShaz ............................................................ 11
detail ................................................................. 12
DX ................................................................. 12
Elements ............................................................ 13
EX1gXK ............................................................... 14
GrpSeqBnds .......................................................... 15
gsd.dens ........................................................... 19
Haybittle ............................................................ 21
IntSurvDiff ........................................................... 23
LanDemets ........................................................... 25
agghaz

Description

Computes the MLE for the model that assumes piecewise constant hazards on intervals defined by a grid of points. One application for example is to calculate monthly hazard rates given numbers of events, numbers at risk and event times reported to the day. Can also handle time to event data stratified on a blocking factor.

Usage

agghaz(t.agg, time, nrisk, nevent)

Arguments

t.agg
   Vector defining intervals upon which the user wants constant hazard rates.

time
   Event times, possibly stratified on a blocking factor into multiple columns, in units that occur in enough numbers per interval specified above. If there is just a single column then it must be in column form (see example below).

nrisk
   Numbers at risk at specified event times

nevent
   Numbers of events at specified event times

Value

time.a
   User supplied left-hand endpoints of intervals of hazard constancy

nrisk.a
   Numbers at risk on specified intervals

nevent.a
   Numbers of events on specified intervals
as.boundaries

Author(s)
Grant Izmirlian <izmirlian@nih.gov>

Examples

library(PwrGSD)
data(lung)
fit.msf <- mysurvfit(Surv(time, I(status==2)) ~ sex, data=lung)

## A single stratum:
with(fit.msf$Table, agghaz(30, time, cbind(nrisk.sex1), cbind(nevent.sex1)))

## Multiple strata--pooled and group 1:
with(fit.msf$Table, agghaz(30, time, cbind(nrisk.sex1+nrisk.sex2,nrisk.sex1),
                         cbind(nevent.sex1+nevent.sex2,nevent.sex1))))

as.boundaries

Convert a "PwrGSD" object to a "boundaries" object

Description
Convert a PwrGSD object to a boundaries object

Usage
as.boundaries(object, ...)

Arguments

object an object of class PwrGSD
...
if object is of class PwrGSD and there are more than one statistic under inves-
tigation, then you may specify an argument stat. The default value is 1, meaning
the first one.

Value
an object of class boundaries. See the documentation for GrpSeqBnds

Author(s)
Grant Izmirlian <izmirlian@nih.gov>

See Also
GrpSeqBnds

Examples

## none as yet
CondPower

CDFOR2LRR  

_Convert CDF Odds Ratio to Logged Relative Risks_

**Description**

Given the values of the baseline hazard and odds ratio of the CDF at a grid of time points find the corresponding logged risk ratio.

**Usage**

CDFOR2LRR(tcut, tmax, h0, CDFOR)

**Arguments**

- **tcut**: Grid of time points (left endpoints)
- **tmax**: The right endpoint of the last interval
- **h0**: Values of the baseline hazard function on given intervals
- **CDFOR**: Values of the odds ratio of the CDF’s on the given intervals

**Value**

An m by 2 matrix, where m=length(tcut), having columns ’tcut’ and logged RR.

**Author(s)**

Grant Izmirlian <izmirlian@nih.gov>

---

CondPower  

_Conditional type I and type II error probabilities given current value of the test statistic_

**Description**

Computes conditional type I and type II error probabilities given current value of the test statistic for monitoring based upon stochastic curtailment. This is now obsolete and included in the functionality of “GrpSeqBnds” and is here for instructional purposes only.

**Usage**

CondPower(Z, frac, drift, drift.end, err.I, sided = 1)
Arguments

- **Z**: Current value of test statistic standardized to unit variance.
- **frac**: Current value of the information fraction (variance fraction).
- **drift**: Current value of the drift, i.e. the expected value of the test statistic normalized to have variance equal to the information fraction. Required if you want to compute conditional type II error, otherwise enter 0.
- **drift.end**: Projected value of the drift at the end of the trial.
- **err.I**: Overall (total) type I error probability
- **sided**: Enter 1 or 2 for sided-ness of the test.

Value

A named numeric vector containing the two components “Pr.cond.typeIerr” and “Pr.cond.typeIIerr”

Author(s)

Grant Izmirlian <izmirlian@nih.gov>

References


See Also

GrpSeqBnds

Examples

```r
## None as yet
```

---

**cpd.PwrGSD**  
*Create a skeleton compound PwrGSD object*

**Description**

Given a user defined indexing dataframe as its only argument, creates a skeleton compound PwrGSD object having a component Elements, a list of PwrGSD objects, of length equal to the number of rows in the indexing dataframe

**Usage**

```r
cpd.PwrGSD(descr)
```
Arguments

descr  A dataframe of a number of rows equal to the length of the resulting list, Elements, of PwrGSD objects. The user defines the mapping between rows of descr and components of Elements and uses it to set up a loop over scenarios. There are several S3 classes and methods for example plot.cpd.PwrGSD, which exploit this mapping between characteristics of a run and the rows of descr for subsetting and constructing conditioned plots. See the example below.

Value

An object of class cpd.PwrGSD containing elements:

date  the POSIX date that the object was created--its quite useful
Elements  a list of length equal to the number of rows of descr which will later contain objects of class PwrGSD
descr  a copy of the indexing dataframe argument for use in navigating the compound object in subsequent calls to other functions such as the related plot method, and the subset extractor, Elements

Note

A cpd.PwrGSD object essentially a list of PwrGSD objects that a user may set up in order to investigate the space of possible trial scenarios, test statistics, and boundary construction options. One could store a list of results without appealing at all to these internal indexing capabilities. The advantage of setting up a cpd.PwrGSD object is the nice summarization functionality provided, for example the plot method for the cpd.PwrGSD class.

The key ingredient to (i) the construction of the empty object, (ii) and summarizing the results in tabular or plotted form via its manipulation in subsequent function calls, is the indexing dataset, descr (for description). The correspondence between rows of descr and elements in the list of PwrGSD objects is purposely left very loose. In the example outlined below, the user creates a “base case” call to PwrGSD and then decides which quantities in this “base case” call to vary in order to navigate the space of possible trial scenarios, monitoring statistics and boundary construction methods. Next, for each one of these settings being varied, a variable with levels that determine each possible setting is created. The dataset descr is created with one line corresponding to each combination of the selection variables so created. In order to ensure that there is 1-1 correspondence between the order of the rows in descr and the order in the list Elements of PwrGSD objects, the user carries out the computation in a loop over rows of descr in which the values of the selection variables in each given row of descr are used to create the corresponding component of Elements via an update the “base case” call.

Author(s)

Grant Izmirlian <izmirlian@nih.gov>

See Also

Elements, plot.cpd.PwrGSD and Power
Examples

## don't worry--these examples are guaranteed to work,
## it's just inconvenient for the package checker
## Not run:
  library(PwrGSD)

## In order to set up a compound object of class 'cpd.PwrGSD'
## we first construct a base case: a two arm trial randomized in just
## under eight years with a maximum of 20 years of follow-up.
## We compute power at a specific alternative, 'rhaz', under
## an interim analysis plan with roughly one annual analysis, some
## crossover between intervention and control arms, with Efficacy
## and futility boundaries constructed via the Lan-Demets procedure
## with O'Brien-Fleming spending on the hybrid scale. Investigate
## the behavior of three weighted log-rank statistics.

test.example <-
  PwrGSD(EfficacyBoundary = LanDemets(alpha = 0.05, spending = ObrienFleming),
  FutilityBoundary = LanDemets(alpha = 0.1, spending = ObrienFleming),
  RR.Futility = 0.82, sided="1<",method="A",accru =7.73, accrat =9818.65,
  tlook =c(7.14, 8.14, 9.14, 10.14, 10.64, 11.15, 12.14, 13.14,
  tcut0 =0:19, h0 =c(3.73e-04, 7.45e-04, 1.49e-03,
  rep(1.49e-03, 15)),
  tcut1 =0:19, rhaz =c(1, 0.9125, 0.8688, 0.7814, 0.6941,
  0.6943, 0.6072, 0.5202, 0.4332, 0.6520, 0.6524, 0.6527, 0.6530, 0.6534, 0.6537,
  0.6541, 0.6544, 0.6547, 0.6551, 0.6554),
  tcutc0 =0:19, hc0 =c(3.73e-04, 7.45e-04, 1.49e-03,
  rep(1.49e-03, 15)),
  tcutc1 =0:19, hc1 =c(3.73e-04, 7.45e-04, 1.49e-03,
  rep(1.49e-03, 15)),
  tcutd0B =c(0, 13), hd0B =c(0.04777, 0),
  tcutd1B =0:6, hd1B =c(0.1109, 0.1381, 0.1485, 0.1637, 0.2446,
  0.2497, 0),
  noncompliance =crossover, gradual =TRUE,
  WtFun =c("FH", "SFH", "Ramp"),
  ppar =c(0, 1, 0, 1, 10, 10))

## we will construct a variety of alternate hypotheses relative to the
## base case specified above

rhaz <-
  c(1, 0.9125, 0.8688, 0.7814, 0.6941, 0.6943, 0.6072, 0.5202, 0.4332,
  0.6520, 0.6524, 0.6527, 0.6530, 0.6534, 0.6537, 0.6541, 0.6544,
  0.6547, 0.6551, 0.6554)

max.effect <- 0.80 + 0.05*(0:8)
n.me <- length(max.effect)

## we will also vary extent of censoring relative to the base case
## specified above
hc <- c(rep(0.0105, 2), rep(0.0209, 3), rep(0.0419, 15))
cens.amt <- 0.75 + 0.25*(0:2)
n.ca <- length(cens.amt)

## we may also wish to compare the Lan-Demets/O'Brien-Fleming efficacy
## boundary with a Lan-Demets/linear spending boundary

Eff.bound.choice <- 1:2
ebc.nms <- c("LanDemets(alpha=0.05, spending=ObrienFleming)",
           "LanDemets(alpha=0.05, spending=Pow(1))")
n.ec <- length(Eff.bound.choice)

## The following line creates the indexing dataframe, 'descr', with one
## line for each possible combination of the selection variables we've
## created.

descr <- as.data.frame(
    cbind(Eff.bound.choice=rep(Eff.bound.choice, each=n.ca*n.me),
          cens.amt=rep(rep(cens.amt, each=n.me), n.ec),
          max.effect=rep(max.effect, n.ec*n.ca)))
descr$Eff.bound.choice <- ebc.nms[descr$Eff.bound.choice]

## Now descr contains one row for each combination of the levels of
## the user defined selection variables, 'Eff.bound.choice',
## 'max.effect' and 'cens.amt'. Keep in mind that the names and number
## of these variables is arbitrary. Next we create a skeleton
## 'cpd.PwrGSD' object with a call to the function 'cpd.PwrGSD' with
## argument 'descr'

test.example.set <- cpd.PwrGSD(descr)

## Now, the newly created object, of class 'cpd.PwrGSD', contains
## an element 'descr', a component 'date', the date created
## and a component 'Elements', an empty list of length equal
## to the number of rows in 'descr'. Next we do the computation in
## a loop over the rows of 'descr'.

n.descr <- nrow(descr)

for(k in 1:n.descr){
    ## First, we copy the original call to the current call,
    ## 'Elements[[k]]$call'

    test.example.set$Elements[[k]]$call <- test.example$call

    ## Use the efficacy boundary choice in the kth row of 'descr'
    ## to set the efficacy boundary choice in the current call
```r
test.example.set$Elements[[k]]$call$EfficacyBoundary <-
parse(text=as.character(descr[k,"Eff.bound.choice"]))[[1]]

## Derive the 'rhaz' defined by the selection variable "max.effect"
## in the kth row of `descr` and use this to set the 'rhaz'
## components of the current call

test.example.set$Elements[[k]]$call$rhaz <-
exp(descr[k,"max.effect"] * log(rhaz))

## Derive the censoring components from the selection variable
## "cens.amt" in the kth row of `descr` and place that result
## into the current call

test.example.set$Elements[[k]]$call$hc0 <-
test.example.set$Elements[[k]]$call$hc1 <- descr[k, "cens.amt"] * hc

## Now the current call corresponds exactly to the selection
## variable values in row 'k' of `descr`. The computation is
## done by calling 'update'

test.example.set$Elements[[k]] <- update(test.example.set$Elements[[k]])
cat(k/n.descr, "$\backslash\text{r}\backslash\text{r}\")

## We can create a new 'cpd.PwrGSD' object by subsetting on
## the selection variables in `descr`

Elements(test.example.set,  
subset=(substring(Eff.bound.choice, 32, 34)=="Obr" & 
max.effect >= 1))

## or we can plot the results -- see the help under `plot.cpd.PwrGSD`

plot(test.example.set, formula = ~ max.effect | stat * cens.amt,
subset=(substring(Eff.bound.choice, 32, 34)=="Obr"))

plot(test.example.set, formula = ~ max.effect | stat * cens.amt,
subset=(substring(Eff.bound.choice, 32, 34)=="Pow"))

## Notice the appearance of the selection variable 'stat' which was
## not defined in the dataset `descr`.

## Recall that each single "PwrGSD" object can contain results
## for a list of test statistics, as in the example shown here where
## we have results on three statistics per component of `Elements`.
## For this reason the variable 'stat' can be also be referenced in
## the `subset` or `formula` arguments of calls to this `plot` method,
## and in the `subset` argument of the function `Power` shown below.

## The function `Power` is used to convert the 'cpd.PwrGSD' object
## into a dataframe, stacked by rows of `descr` and by `stat`
```

## Cumulative-risk ratios to risk ratios

**Description**

Given a vector of cumulative-risk ratios, computes risk ratios

**Usage**

```r
CRRtoRR(CRR, DT, h = NULL)
```

**Arguments**

- `CRR` vector of cumulative risk ratios of length \(m\)
- `DT` vector of time increments upon which the cumulative ratios represent. For example if the hazard takes values \(h_1, h_2, \ldots, h_m\) on the intervals \([t_1, t_2), [t_2, t_3), \ldots, [t_m, t_m+1)\) then \(DT\) will be \(c(t_2-t_1, t_3-t_2, \ldots, t_{m+1}-t_m)\)
- `h` The hazard in the reference arm, of length \(m\)

**Value**

The vector of risk ratios at the \(m\) time points

**Author(s)**

Grant Izmirlian <izmirlian@nih.gov>

**Examples**

```r
## none as yet
```


Description

Given the cutpoints at which the hazard is to be constant, the values taken by the calendar year rates and the calendar time offset from the start of the trial at which randomization ended, this function converts to time on study rates, assuming uniform accrual.

Usage

\[\text{CY2TOShaz}(tcut, t.eor, m, verbose = \text{FALSE})\]

Arguments

- **tcut**: Left hand endpoints of intervals on which time on study hazard is taken to be constant
- **t.eor**: Time offset from the beginning of the trial at which randomization ended
- **m**: Annual calendar time rates
- **verbose**: do you want to see a lot of debugging info—defaults to FALSE

Value

- **hazard** = h, **table** = attr(obj., "tbl")
  - **hazard**: time on study hazard values taken on intervals specified by the argument tcut
  - **table**: a table containing the observed and fitted values

Author(s)

Grant Izmirlian <izmirlian@nih.gov>

Examples

```r
## none as yet
```
detail

Function to extract the 'detail' component from a PwrGSD object

Description

Extracts the 'detail' component from an object of class PwrGSD

Usage

detail(obj)

Arguments

obj An object of class PwrGSD returned from the function PwrGSD or a component of the list returned by the function cpd.PwrGSD

Value

The 'detail' component of the object. For the Asymptotic method, this will be most of the quantities involved in the computation, the input parameters such as the various incidence rates, cross over rates etc, as well as intermediate computations such as the drift function variance function as well. For the simulation method, some of these are returned an in addition, the simulated event histories.

Author(s)

Grant Izmirlian <izmirlian at nih dot gov>

DX

A utility function for forming differences

Description

DX(x) returns c(x[1], diff(x))

Usage

DX(x)

Arguments

x A grid of time points (increasing)

Value

DX(x) returns c(x[1], diff(x))
Create a subset of a "cpd.PwrGSD" object

Description
Create a subset of a cpd.PwrGSD object

Usage

Elements(object, subset, na.action = na.pass)

Arguments

- **object**: an object of class cpd.PwrGSD
- **subset**: you may extract a subset via a logical expression in the variables of the index dataframe, descr
- **na.action**: a method for handling NA values in the variables in the subset expression.

Value

an object of class cpd.PwrGSD. See help on that topic for details.

Author(s)

Grant Izmirlian <izmirlian@nih.gov>

See Also

cpd.PwrGSD and PwrGSD

Examples

```r
## See the 'cpd.PwrGSD' example
```
A function for computing the bias adjusted point estimate for a statistic observed to cross the efficacy boundary.

Description

A function for computing the bias adjusted point estimate for a statistic, on the Brownian scale, observed to cross the efficacy boundary.

Usage

EX1gXK(xk, b.eff, frac)

Arguments

- **xk**: The observed value of the statistic, on the “Brownian” scale.
- **b.eff**: Efficacy boundary points at current and prior analyses
- **frac**: Information fraction at current and prior analyses

Value

Returns the expected value of $X_1$ given $X_K$, which is the bias adjusted point estimate

Note

This works for the unweighted, proportional hazards case, but also works in the case of the weighted log-rank statistic when we assume the chosen weights are proportional to the true shape.

Author(s)

Grant Izmirlian <izmirlig@mail.nih.gov>

References


Izmirlian, G. (2014). Estimation of the relative risk following group sequential procedure based upon the weighted log-rank statistic. Statistics and its Interface 00 00–00

See Also

gsd.dens
Examples

# if Z.K = U_K/V_K^0.5 is the log-rank statistic on the standard normal
# scale, then we obtain an estimate of the logged relative risk as follows
# Suppose we've stopped at analysis number K=4, and Z.K = 2.5
# suppose the end of trial variance of the log-rank statistic
# (specified in design and used to compute 'frac') is V.end = 100

K <- 4
Z.K <- 2.5
V.end <- 100

# Information fraction
frac <- c(0.15, 0.37, 0.64, 0.76)

# Efficacy Boundary
gsb <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(spending=ObrienFleming, alpha=0.05))

# Efficacy boundary points
be <- gsb$table[,"b.e"]

# Brownian scale
X.K <- Z.K*frac[K]

# expected value of X_1 given X_K
ex1gxk <- EX1gXK(X.K, be, frac)

# Crude estimate of logged relative risk
X.K/(frac[K]*V.end^0.5)

# Bias adjusted estimate of logged relative risk
ex1gxk/(frac[1]*V.end^0.5)

GrpSeqBnds

Computes efficacy and futility boundaries

Description

This computes efficacy and futility boundaries for interim analysis and sequential designs. Two sided symmetric efficacy boundaries can be computed by specifying half of the intended total type I error probability in the argument, Alpha.Efficacy. Otherwise, especially in the case of efficacy and futility bounds only one sided boundaries are currently computed. The computation allows for two different time scales—one must be the variance ratio, and the second can be a user chosen increasing scale beginning with 0 that takes the value 1 at the conclusion of the trial.

Usage

GrpSeqBnds(EfficacyBoundary = LanDemets(alpha = 0.05, spending = ObrienFleming),
FutilityBoundary = LanDemets(alpha = 0.1, spending = ObrienFleming),
NonBindingFutility = TRUE, frac, frac.ii = NULL, drift = NULL)
Arguments

EfficacyBoundary
This specifies the method used to construct the efficacy boundary. The available choices are:

(i) 'Lan-Demets(alpha=<total type I error>, spending=<spending function>). The Lan-Demets method is based upon a error probability spending approach. The spending function can be set to ObrienFleming, Pocock, or Power(rho), where rho is the the power argument for the power spending function: rho=3 is roughly equivalent to the O'Brien-Fleming spending function and smaller powers result in a less conservative spending function.

(ii) 'Haybittle(alpha=<total type I error>, b.Haybittle=<user specified boundary point>). The Haybittle approach is conceptually the simplest of all methods for efficacy boundary construction. However, as it spends nearly no alpha until the end, is for all practical purposes equivalent to a single analysis design and to be considered overly conservative. This method sets all the boundary points equal to b.Haybittle, a user specified value (try 3) for all analyses except the last, which is calculated so as to result in the total type I error, set with the argument alpha.

(iii) 'SC(be.end=<efficacy boundary point at trial end>, prob=<threshold for conditional type I error for efficacy stopping>). The stochastic curtailment method is based upon the conditional probability of type I error given the current value of the statistic. Under this method, a sequence of boundary points on the standard normal scale (as are boundary points under all other methods) is calculated so that the total probability of type I error is maintained. This is done by considering the joint probabilities of continuing to the current analysis and then exceeding the threshold at the current analysis. A good value for the threshold value for the conditional type I error, prob, is 0.90 or greater.

(iv) 'User supplied boundary points in the form c(b1,b2,b3,...,b_m), where m is the number of looks.

FutilityBoundary
This specifies the method used to construct the futility boundary. The available choices are:

(i) 'Lan-Demets(alpha=<total type II error>, spending=<spending function>). The Lan-Demets method is based upon a error probability spending approach. The spending function can be set to ObrienFleming, Pocock, or Power(rho), where rho is the the power argument for the power spending function: rho=3 is roughly equivalent to the O'Brien-Fleming spending function and smaller powers result in a less conservative spending function.

NOTE: 'there is no implementation of the Haybittle method for futility boundary construction. Given that the futility boundary depends upon values of the drift function, this method doesn’t apply.

(ii) 'SC(be.end=<efficacy boundary point at trial end>, prob=<threshold for conditional type II error for futility stopping>, drift.end=<projected drift at end of trial>). The stochastic curtailment method is based upon the conditional probability of type II error given the current value of the statistic. Under this method, a sequence of boundary points on the standard normal scale (as are boundary points under all other methods) is calculated so that the total proba-
bility of type II error, is maintained. This is done by considering the joint prob-
abilities of continuing to the current analysis and then exceeding the threshold
at the current analysis. A good value for the threshold value for the conditional
type I error, prob is 0.90 or greater.

(iii) 'User supplied boundary points in the form c(b1, b2, b3, ... , b_m), where
m is the number of looks.

**NonBindingFutility**

When using a futility boundary and this is set to 'TRUE', the efficacy boundary
will be constructed in the absence of the futility boundary, and then the futility
boundary will be constructed given the resulting efficacy boundary. This results
in a more conservative efficacy boundary with true type I error less than the
nominal level. This is recommended due to the fact that futility crossings are
viewed by DSMB's with much less gravity than an efficacy crossing and as
such, the consensus is that efficacy bounds should not be discounted towards the
null hypothesis because of paths which cross a futility boundary. Default value
is 'TRUE'.

**frac**
The variance ratio. If the end of trial variance is unknown then normalize all pre-
vious variances by the current variance. In this case you must specify a second
scale that is monotone increasing from 0 to 1 at the end of the trial. Required.

**frac.ii**
The second information scale that is used for type I and type II error probability
spending. Optional (see above)

**drift**
The drift function of the underlying brownian motion, which is the expected
value under the design alternative of the un-normalized weighted log-rank statis-
tic, then normalized to have variance one when the variance ratio equals 1. See
the examples below.

**Value**

An object of class boundaries with components: "table" "frac" "frac.ii" "drift" "call"

call           The call that produced the returned results.
frac           The vector of variance ratios.
frac.ii        The vector of information ratios for type I and type II error probability spending,
               which differs from the above if the user sets the argument frac.ii to a second
               scale as mentioned above.

drift          The drift vector that is required as an argument when futility boundaries are
calculated.

**table**

A matrix with components

'frac 'The information ratio for type I and type II error probability spending.
'b.f 'The calculated futility boundary (if requested).
'alpha.f 'The type II error probability spent at that analysis (if doing futility
bounds).
'cum-alpha.f 'Cumulative sum of alpha.f (if doing futility bounds).
'b.e 'The calculated efficacy boundary.
'alpha.e 'The type I error probability spent at that analysis.
'cum-alpha.e 'Cumulative sum of alpha.e.
Author(s)

Grant Izmirlian <izmirlian@nih.gov>

References


Izmirlian G. (2014). Estimation of the relative risk following group sequential procedure based upon the weighted log-rank statistic. Statistics and its Interface 7(1), 27-42

See Also

PwrGSD

Examples

## NOTE: In an unweighted analysis, the variance ratios and event ratios
## are the same, whereas in a weighted analysis, they are quite different.
##
## For example, in a trial with 7 or so years of accrual and maximum follow-up of 20 years
## using the stopped Fleming-Harrington weights, `WtFun` = "SFH", with parameters
## `ppar` = c(0, 1, 10) we might get the following vector of variance ratios:

frac <- c(0.00695655, 0.01444565, 0.02682463, 0.04641363, 0.0585665,
           0.07614902, 0.1135391, 0.168252, 0.2336901, 0.3186155, 0.4164776,
           0.5352199, 0.670739, 0.8246061, 1)

## and the following vector of event ratios:

frac.ii <- c(0.1494354, 0.1972965, 0.2625075, 0.3274323, 0.3519184, 0.40231,
             0.4673837, 0.5579035, 0.6080742, 0.6982293, 0.7671917, 0.8195019,
             0.9045182, 0.9515884, 1)

## and the following drift under a given alternative hypothesis

drift <- c(0.06214444, 0.1061856, 0.1731267, 0.2641265, 0.3105231, 0.3836636,
           0.5117394, 0.6918584, 0.8657705, 1.091984, 1.311094, 1.538582,
           1.81346, 2.081775, 2.345386)

## JUST ONE SIDED EFFICACY BOUNDARY
## In this call, we calculate a one sided efficacy boundary at each of 15 analyses
## which will occur at the given (known) variance ratios, and we use the variance
## ratio for type I error probability spending, with a total type I error probability
gsb.all.just.eff <- GrpSeqBnds(frac=frac,
   EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming))

gsb.all.eff.fut <- GrpSeqBnds(frac=frac,
   EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming),
   FutilityBoundary=LanDemets(alpha=0.10, spending=ObrienFleming),
   drift=drift)

## Now suppose that we are performing the 7th interim analysis. We don't know what the variance
## will be at the end of the trial, so we normalize variances of the current and previous
## statistics by the variance of the current statistic. This is equivalent to the following
## length 7 vector of variance ratios:
frac7 <- frac[1:7]/frac[7]

## To proceed under the “unknown variance at end of trial” case, we must use a second
## scale for spending type I and II error probability. Unlike the above scale
## which is renormalized at each analysis to have value 1 at the current analysis, the
## alpha spending scale must be monotone increasing and attain the value 1 only at the
## end of the trial. A natural choice is the event ratio, which is known in advance if
## the trial is run until a required number of events is obtained, a so called
## maximum information trial:
frac7.ii <- frac.ii[1:7]

## the first seven values of the drift function
drift7 <- drift[1:7]/frac[7]^0.5

gsb.1st7.eff.fut <- GrpSeqBnds(frac7, frac.ii=frac7.ii,
   EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming),
   FutilityBoundary=LanDemets(alpha=0.10, spending=ObrienFleming),
   drift=drift7)

## Of course there are other options not covered in these examples but this should get you
## started

gsd.dens

A function for computing the probability density for the group sequentially monitored test statistic.
Description

A function for computing the probability density for a sequentially monitored test. This is the joint density, in the rejection region, of \((X_K, K)\), where \(X_K\) is the observed value of the test statistic upon efficacy boundary crossing, and \(K\) is the analysis number at which the efficacy boundary was crossed.

Usage

\[
gsd.dens(x, \text{frac} = \text{NULL}, \text{scale} = \text{"Standard"})
\]

Arguments

- **x**: The main argument, \(x\), is either an object of class “boundaries” or a numeric vector. If it is of class “boundaries” then no other arguments are required. If it is a numeric vector then the \(\text{frac}\) argument must be specified. See below. In this case, \(x\) will be the observed values of the statistic at the current and all prior analyses, either on the standard normal scale (the default) or on the “Brownian” scale. For “Brownian” scale, set argument \(\text{scale}\) to “Brownian”.
- **frac**: Required only when the main argument, \(x\), is a numeric vector, and must be a vector of the same length. In this case, \(\text{frac}\) will be the information at the current and all prior interim analyses.
- **scale**: Required only when the main argument, \(x\), is a numeric vector. A switch indicating whether the elements of the numeric vector, \(x\), are specified on the standard normal scale, \(x=\text{"Standard"}\), or on the Brownian scale, \(x=\text{"Brownian"}\).

Value

A list with elements \(x\), \(dF\), \(x1c\), and \(dF1c\):

- **\(x\)**: Node points used in Gaussian quadrature. See examples below.
- **\(dF\)**: Probability mass at each node point. See examples below.
- **\(x1c\)**: Node points in the continuation region at the first analysis.
- **\(dF1c\)**: Probability mass at each node point in the continuation region at the first analysis.

Note

Also used in computation of Rao-Blackwell-ized bias adjusted point estimate for statistic observed to cross the efficacy boundary.

Author(s)

Grant Izmirlian <izmirlig@mail.nih.gov>
References


Izmirlian, G. (2014). Estimation of the relative risk following group sequential procedure based upon the weighted log-rank statistic. Statistics and its Interface 00 00–00

See Also

EX1gXX

Examples

# Information fraction
frac <- c(0.15, 0.37, 0.64, 0.76)

# Efficacy Boundary
gsb <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(spending=ObrienFleming, alpha=0.05))

# To compute the p-value under the stagewise ordering, for an observed value of the monitoring statistic 2.1, crossing the efficacy boundary at the 4th analysis, we do the following
be <- gsb$table[,"b.e"]
be[4] <- 2.1
sum(gsd.dens(be, frac, scale="Standard")$df)

Haybittle

The Haybittle method of Boundary Construction

Description

The function Haybittle is used in calls to the functions GrpSeqBnds and PwrGSD as a possible setting for the argument EfficacyBoundary. NOTE: the Haybittle method is not implemented as a futility boundary method. The Haybittle method is one of four currently available choices (efficacy only), the others being LanDemets, SC (stochastic curtailment), and user specified.

Usage

Haybittle(alpha, b.Haybittle, from = NULL, to = NULL)
Arguments

alpha  The total probability of type I error.
b.Haybittle  User specified efficacy boundary at all but the last analysis.

Details

The Haybittle approach is conceptually the simplest of all methods for efficacy boundary construction. However, as it spends nearly no alpha until the end, is for all practical purposes equivalent to a single analysis design and to be considered overly conservative. This method sets all the boundary points equal to b.Haybittle, a user specified value (try 3) for all analyses except the last, which is calculated so as to result in the total type I error, set with the argument alpha.

Value

An object of class boundary.construction.method which is really a list with the following components. The print method displays the original call.

type  Gives the boundary construction method type, which is the character string "Haybittle"
alpha  The numeric value passed to the argument 'alpha' which is the total probability of type I error.
b.Haybittle  The numeric value passed to the argument 'b.Haybittle' which is the user specified efficacy boundary at all but the last analysis.
from  Description of 'comp2'
to  You’re not using this, right?
call  see above.

Note

The print method returns the call by default

Author(s)

Grant Izmirlian

References

see references under PwrGSD

See Also

LanDemets, SC, GrpSeqBnds, and PwrGSD
## Examples

### example 1: what is the result of calling a Boundary Construction Method function

```r
# A call to 'Haybittle' just returns the call
Haybittle(alpha=0.05, b.Haybittle=3)
```

### It does argument checking...this results in an error

```r
# Not run:
Haybittle(alpha=0.05)
```

### End(Not run)

### but really its value is a list with the a component containing
### the boundary method type, "LanDems", and components for each
### of the arguments.

```r
names(Haybittle(alpha=0.05, b.Haybittle=3))
```

```r
Haybittle(alpha=0.05, b.Haybittle=3)$type
Haybittle(alpha=0.05, b.Haybittle=3)$alpha
Haybittle(alpha=0.05, b.Haybittle=3)$b.Haybittle
Haybittle(alpha=0.05, b.Haybittle=3)$call
```

### example 2: ...But the intended purpose of the spending functions
### is in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

```r
frac <- c(0.07614902,0.1135391,0.168252,0.2336901,0.3186155, 0.4164776,0.5352199,0.670739,0.8246061,1)

test <- GrpSeqBnds(frac=frac, EfficacyBoundary=Haybittle(alpha=0.025, b.Haybittle=3))
```

---

### IntSurvDiff

**Weighted Integrated Survival function test**

**Description**

Computes a two sample weighted integrated survival function log-rank statistic with events weighted according to one of the available weighting function choices

**Usage**

```r
IntSurvDiff(formula =formula(data), data =parent.frame(), WtFun =c("FH", "SFH", "Ramp"), param = c(0, 0), sided = c(2, 1), subset, na.action, w = FALSE)
```

**Arguments**

- `formula`: a formula of the form `Surv(Time,Event) ~ arm` where `arm` is a dichotomous variable with values 0 and 1.
- `data`: a dataframe
WtFun

a selection from the available list: “FH” (Fleming-Harrington), “SFH” (stopped Fleming-Harrington) or “Ramp”. See param in the following line.

param

Weight function parameters. Length and interpretation depends upon the selected value of WtFun:

If WtFun==“FH” then param is a length 2 vector specifying the power of the pooled (across arms) Kaplan-Meier estimate and its complement.

If WtFun==“SFH” then param is a length 3 vector with first two components as in the “FH” case, and third component the time (in the same units as the time to event) at which the “FH” weight function is capped off at its current value.

If WtFun==SFH then param is of length 1 specifying the time (same units as time to event) at which events begin to get equal weight. The “Ramp” weight function is a linearly increasing deterministic weight function which is capped off at 1 at the user specified time.

sided

One or Two sided test? Set to 1 or 2

subset

Analysis can be applied to a subset of the dataframe based upon a logical expression in its variables

na.action

Method for handling NA values in the covariate, arm

w

currently no effect

Value

An object of class survtest containing components

pn
sample size

wtyp
internal representation of the WtFun argument

par
internal representation of the param argument

time
unique times of events across all arms

nrisk
Number at risk across all arms at each event time

nrisk1
Number at risk in the experimental arm at each event time

nevent
Number of events across all arms at each event time

nevent1
Number of events in the experimental arm at each event time

wt
Values of the weight function at each event time

pntimes
Number of event times

stat
The un-normalized weighted log-rank statistic, i.e. the summed weighted observed minus expected differences at each event time

var
Variance estimate for the above

pu0
person units of follow-up time in the control arm

pu1
person units of follow-up time in the intervention arm

n0
events in the control arm

n1
events in the intervention arm

n
sample size, same as pn

call
the call that created the object
Author(s)
Grant Izmirlian <izmirlian@nih.gov

References

See Also
wtdlogrank

Examples

```r
library(PwrGSD)
data(lung)
fit.isd<-IntSurvDiff(Surv(time,I(status==2))~I(sex==2), data=lung, WtFun="SFH", param=c(0,1,300))
```

---

**LanDemets**  
*The Lan-Demets method of Boundary Construction*

Description
The function LanDemets is used in calls to the functions GrpSeqBnds and PwrGSD as a possible setting for the arguments EfficacyBoundary and FutilityBoundary, in specification of the method whereby efficacy and or futility boundaries are to be constructed. The Lan-Demets method is one of four currently available choices, the others being SC (stochastic curtailment), Haybittle (efficacy only) and user specified.

Usage

```r
LanDemets(alpha, spending, from = NULL, to = NULL)
```

Arguments

- **alpha**: If LanDemets is used to specify the EfficacyBoundary then the argument alpha is the total probability of type I error. If LanDemets is used to specify the FutilityBoundary then the argument alpha is the total probability of type II error.
- **spending**: Specify the alpha spending function. Set this to ObrienFleming, Pow(rho=<x>), or Pocock. See help files for these spending functions.
- **from**: WARNING EXPERIMENTAL: you can actually construct boundaries via a hybrid of the 3 boundary construction methods, LanDemets, SC, and 'user specified'. When using a hybrid boundary, set the argument EfficacyBoundary or FutilityBoundary respectively, to a list with components LanDemets, SC, or user specified numbers. In the former two cases, from and to are used in
LanDemets

and also in SC to stipulate how many interim analyses they are in effect. See the help for GrpSeqBnds and PwrGSD

to See above.

Details

The cornerstone of the Lan-Demets method is that the amount of alpha (type I or II error probability) that is "spent" at a given interim analysis is determined via a user specified "spending function". A spending function is a monotone increasing mapping on (0,1) with range (0,alpha). The 'alpha' spent at a given analysis is determined by the increment in the values of the spending function at the current and at the most recent information fractions.

Value

An object of class boundary.construction.method which is really a list with the following components. The print method displays the original call.

type Gives the boundary construction method type, which is the character string "LanDemets"

alpha The numeric value passed to the argument 'alpha' which is the total probability of type I (efficacy) or type II (futility) error.

spending The spending function that was passed to the argument 'spending'. Note that this will be of class 'name' for 'ObrienFleming' and 'Pocock', but will be of class 'function' for 'Pow'

from The numeric value passed to the argument 'from'. See above.

to The numeric value passed to the argument 'to'. See above.

call returns the call

Note

The print method returns the call by default

Author(s)

Grant Izmirlian

References

see references under PwrGSD

See Also

SC, ObrienFleming, Pow, Pocock, GrpSeqBnds, and PwrGSD
Examples

## example 1: what is the result of calling a Boundary Construction Method function
## A call to 'LanDemets' just returns the call
LanDemets(alpha=0.05, spending=ObrienFleming)

## It does argument checking...this results in an error
## Not run:
LanDemets(alpha=0.05)

## End(Not run)

## but really its value is a list with the a component containing
## the boundary method type, "LanDemets", and components for each
## of the arguments.
names(LanDemets(alpha=0.05, spending=ObrienFleming))

LanDemets(alpha=0.05, spending=ObrienFleming)$type
LanDemets(alpha=0.05, spending=ObrienFleming)$alpha
LanDemets(alpha=0.05, spending=ObrienFleming)$spending
class(LanDemets(alpha=0.05, spending=ObrienFleming)$spending)
LanDemets(alpha=0.05, spending=Pocock)$spending
class(LanDemets(alpha=0.05, spending=Pocock)$spending)
LanDemets(alpha=0.05, spending=ObrienFleming)$call

## example 2: ...But the intended purpose of the spending functions is
## in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

frac <- c(0.07614902,0.1135391,0.168252,0.2336901,0.3186155,
          0.4164776,0.5352199,0.670739,0.8246061,1)
drift <- c(0.3836636,0.5117394,0.6918584,0.8657705,1.091984,
          1.311094,1.538582,1.818346,2.081775,2.345386)
test <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming),
                   FutilityBoundary=LanDemets(alpha=0.10, spending=Pocock),
                   drift=drift)

lookup

Lookup values for a piecewise constant function

Description

Given the values and lefthand endpoints for intervals of constancy, lookup values of the function at
arbitrary values of the independent variable.

Usage

lookup(xgrid, ygrid, x, y0 = 0)
**Arguments**

- **xgrid**: Lefthand endpoints of intervals of constancy
- **ygrid**: Values on these intervals, of same length as xgrid
- **x**: Input vector of arbitrary independent variables.
- **y0**: Value to be returned for values of x that are smaller than min(xgrid).

**Value**

- Describe the value returned. If it is a LIST, use
  - comp1: Description of 'comp1'
  - comp2: Description of 'comp2'

**Author(s)**

Grant Izmirlian <izmirlian@nih.gov>

**Examples**

```r
## none as yet
```

---

**lung**  
*Mayo Clinic Lung Cancer Data*

**Description**

Survival in patients with lung cancer at Mayo Clinic. Performance scores rate how well the patient can perform usual daily activities.

**Usage**

```r
data(lung)
```

**Format**

- **inst**: Institution code
- **time**: Survival time in days
- **status**: Censoring status 1=censored, 2=dead
- **age**: Age in years
- **sex**: Male=1 Female=2
- **ph.ecog**: ECOG performance score (0=good 5=dead)
- **ph.karno**: Karnofsky performance score (bad=0-good=100) rated by physician
- **pat.karno**: Karnofsky performance score rated by patient
- **meal.cal**: Calories consumed at meals
- **wt.loss**: Weight loss in last six months
mystack

Source
Terry Therneau

mystack Stack a dataset

Description
Given a dataframe containing one or more variables named with a common prefix, this function creates a stacked dataset with one set of observed values of the variables (in order of occurrence) per line.

Usage
mystack(object, fu.vars, create.idvar = FALSE)

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a dataframe containing one or more variables named with a common prefix</td>
</tr>
<tr>
<td>fu.vars</td>
<td>a list of the unique prefixes</td>
</tr>
<tr>
<td>create.idvar</td>
<td>Do you want to add an ID variable with a common value given to all records resulting from a given input record? Default is FALSE</td>
</tr>
</tbody>
</table>

Value
A stacked dataframe

Author(s)
Grant Izmirlian <izmirlian@nih.gov>

Examples
```r
# none as yet
```
mysurvfit

My Survfit

Description
Computes numbers at risk, numbers of events at each unique event time within levels of a blocking factor

Usage
mysurvfit(formula = formula(data), data = parent.frame(), subset, na.action = na.fail)

Arguments
- formula: Should be a formula of the form Surv(ti,ev) ~ block where block is the blocking factor. It need not be a factor per se but should have relatively few discrete levels. Sorry, no staggered entry allowed at present
- data: a dataframe
- subset: you can subset the analysis via logical expression in variables in the dataframe
- na.action: pass a method for handling NA values in block such as na.omit, or na.fail

Value
A dataframe of 2*NLEV + 1 columns where NLEV is the number of levels of the factor block.
- time: The sorted vector of unique event times from all blocks
- nrisk1: The number at risk in block level 1 at each event time
- nevent1: The number of events in block level 1 at each event time
- ...
- nriskNLEV: The number at risk in block level NLEV at each event time
- neventNLEV: The number of events in block level NLEV at each event time

Author(s)
Grant Izmirlian <izmirlian@nih.gov>

Examples
library(PwrGSD)
data(lung)
fit.msf <- mysurvfit(Surv(time, I(status==2)) ~ sex, data=lung)

fit.msf
## Not run:
plot(fit.msf)
## End(Not run)
Description

Stipulates alpha spending according to the O’Brien-Fleming spending function in the Lan-Demets boundary construction method. Its intended purpose is in constructing calls to GrpSeqBnds and PwrGSD.

Usage

ObrienFleming()

Value

An object of class spending.function which is really a list with the following components. The print method displays the original call.

type Gives the spending function type, which is the character string "ObrienFleming"
call returns the call

Note

The print method returns the call by default

Author(s)

Grant Izmirlian

References

see references under PwrGSD

See Also

LanDemets, Pow, Pocock, GrpSeqBnds, PwrGSD

Examples

```r
## example 1: what is the result of calling a spending function
## A call to 'ObrienFleming' just returns the call
ObrienFleming()

## but really its value is a list with a component named
## 'type' equal to "ObrienFleming" and a component named
## 'call' equal to the call.
names(ObrienFleming)

ObrienFleming()$type
```
ObrienFleming()

## example 2: ...But the intended purpose of the spending functions is
## in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

frac <- c(0.07614902, 0.1135391, 0.168252, 0.2336901, 0.3186155,
          0.4164776, 0.5352199, 0.670739, 0.8246061, 1)
drift <- c(0.3836636, 0.5117394, 0.6918584, 0.8657705, 1.091984,
           1.311094, 1.538582, 1.818346, 2.081775, 2.345386)

test <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming),
                   FutilityBoundary=LanDemets(alpha=0.10, spending=Pocock),
                   drift=drift)

plot.cpd.PwrGSD

## S3 method for class 'cpd.PwrGSD'
plot(x, formula, subset, na.action, ...)

Arguments

x an object of class cpd.PwrGSD

formula a one sided formula of the form ~ effect | f1 or ~ effect | f1 * f2 where
effect, f1, and f2 are variables in the indexing dataframe descr, or the special
variable stat which may be used when there are multiple test statistics per com-
ponent of Elements. See the example in the documentation for cpd.PwrGSD.

subset the plot can be applied to a subset of rows of descr via a logical expression on
its variables in combination with the special variable, stat when applicable.

na.action a na.action method for handling NA values

... other parameters to pass to the R function coplot usually not neccesary

Value

Returns the object, x, invisibly

Note

This processes the cpd.PwrGSD object into a dataframe, stacked on interim looks and then passes
the results to the R function coplot
**Pocock**

**Author(s)**

Abovementioned cpd.PwrGSD processing done by Grant Izmirlian <izmirlian@nih.gov>

**References**


**See Also**

cpd.PwrGSD Power and Elements

**Examples**

```r
## See the example in the 'cpd.PwrGSD' documentation
```

---

**Pocock**

*The Pocock Alpha Spending Function*

**Description**

Stipulates alpha spending according to the Pocock spending function in the Lan-Demets boundary construction method. Its intended purpose is in constructing calls to GrpSeqBnds and PwrGSD.

**Usage**

```r
Pocock()
```

**Value**

An object of class spending.function

- **type**
  
  Gives the spending function type, which is the character string "Pocock"

- **call**
  
  returns the call

**Note**

The print method returns the call by default

**Author(s)**

Grant Izmirlian

**References**

See references under PwrGSD
See Also

LanDemets, ObrienFleming, Pow, GrpSeqBnds, PwrGSD

Examples

## example 1: what is the result of calling a spending function

## A call to 'Pocock' just returns the call
Pocock()

## but really its value is a list with a component named
## 'type' equal to "Pocock" and a component named
## 'call' equal to the call.
names(Pocock)

Pocock()$type

Pocock()$call

## example 2: ...But the intended purpose of the spending functions is
## in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

frac <- c(0.07614902,0.1135391,0.168252,0.2336901,0.3186155,
          0.4164776,0.5352199,0.670739,0.8246061,1)

drift <- c(0.3836636,0.5117394,0.6918584,0.8657705,1.091984,
          1.311094,1.538582,1.818346,2.081775,2.345386)

test <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(alpha=0.05, spending=Pocock),
                    FutilityBoundary=LanDemets(alpha=0.10, spending=ObrienFleming),
                    drift=drift)

---

Pow The Wang-Tsiatis Power Alpha Spending Function

Description

Stipulates alpha spending according to the Wang-Tsiatis Power function in the Lan-Demets boundary construction method. Its intended purpose is in constructing calls to GrpSeqBnds and PwrGSD.

Usage

Pow(rho)

Arguments

rho The exponent for the Wang-Tsiatis power spending function
Details
Larger rho results in more conservative boundaries. rho=3 is roughly equivalent to Obrien-Fleming spending. rho=1 spends alpha linearly in the information fraction.

Value
An object of class spending.function which is really a list with the following components. The print method displays the original call.

type: Gives the spending function type, which is the character string "Pow"
rho: the numeric value passed to the single argument, rho
call: returns the call

Note
The print method returns the call by default

Author(s)
Grant Izmirlian

References
see references under PwrGSD

See Also
LanDemets, ObrienFleming, Pocock, GrpSeqBnds, PwrGSD

Examples
```r
## example 1: what is the result of calling a spending function
## A call to 'Pow' just returns the call
Pow(rho=2)

## It does argument checking...the following results in an error:
## Not run:
Pow()

## End(Not run)

## it doesn't matter whether the argument is named or not,
## either produces the same result
Pow(2)

## but really its value is a list with a component named
## 'type' equal to "Pow", a component named 'rho' equal
## to the numeric value passed to the single argument 'rho'
## and a component named 'call' equal to the call.
names(Pow(rho=2))
```
## example 2: ...But the intended purpose of the spending functions is
## in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

```r
frac <- c(0.07614902, 0.1135391, 0.168252, 0.2336901, 0.3186155,
          0.4164776, 0.5352199, 0.670739, 0.8246061, 1)
drift <- c(0.3836636, 0.5117394, 0.6918584, 0.8657705, 1.091984,
           1.311094, 1.538582, 1.818346, 2.081775, 2.345386)
test <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(alpha=0.05, spending=Pow(2)),
                   FutilityBoundary=LanDemets(alpha=0.10, spending=ObrienFleming),
                   drift=drift)
```

### Description

The function ‘Power’ is used to summarize the ‘cpd.PwrGSD’ object into a dataframe containing power and type II error, summed over analysis times. The data frame is stacked by rows of ‘descr’ and by ‘stat’ (if there are multiple statistics being profiled per each component of ‘Elements’), for generating tables or performing other computations.

### Usage

```r
Power(object, subset, nlook.ind = NULL)
```

### Arguments

- `object`: an object of class `cpd.PwrGSD`
- `subset`: you may extract a subset via a logical expression in the variables of the index dataframe, `descr`
- `nlook.ind`: (optional) a vector containing a subset of the indices of analysis times over which the sum is formed. Use this for example if you want to know the probability of stopping by the kth analysis under an unfavorable alternative. Set `nlook.ind` to `1:k`

### Value

A dataframe, stacked by rows of ‘descr’ and then by choices of ‘stat’
Author(s)

Grant Izmirlian <izmirlian@nih.gov>

See Also

cpd.PwrGSD and PwrGSD

Examples

```r
## See the `cpd.PwrGSD` example
```

---

### Description

Derives power in a two arm clinical trial under a group sequential design. Allows for arbitrary number of interim analyses, arbitrary specification of arm-0/arm-1 time to event distributions (via survival or hazard), arm-0/arm-1 censoring distribution, provisions for two types of continuous time non-compliance according to arm-0/arm-1 rate followed by switch to new hazard rate. Allows for analyses using (I) weighted log-rank statistic, with weighting function (a) a member of the Flemming-Harrington G-Rho class, or (b) a stopped version thereof, or (c) the ramp-plateau deterministic weights, or (II) the integrated survival distance (currently under method=="S" without futility only). Stopping boundaries are computed via the Lan-Demets method, Haybittle method, converted from the stochastic curtailment procedure, or be completely specified by the user. The Lan-Demets boundaries can be constructed using either O'Brien-Flemming, Pocock or Wang-Tsiatis power alpha-spending. The C kernel is readily extensible, and further options will become available in the near future.

### Usage

```r
PwrGSD(EfficacyBoundary = LanDemets(alpha = 0.05, spending = ObrienFleming),
     FutilityBoundary = LanDemets(alpha = 0.1, spending = ObrienFleming),
     NonBindingFutility = TRUE, sided = c("2>", "2<", "1>", "1<"),
     method = c("S", "A"), accru, accrat, tlook,
     tcut0 = NULL, h0 = NULL, s0 = NULL, tcut1 = NULL,
     rhaz = NULL, h1 = NULL, s1 = NULL, tcutc0 = NULL, hc0 = NULL,
     sc0 = NULL, tcutc1 = NULL, hc1 = NULL, sc1 = NULL, tcutd0A = NULL,
     hd0A = NULL, sd0A = NULL, tcutd0B = NULL, hd0B = NULL, sd0B = NULL,
     tcutd1A = NULL, hd1A = NULL, sd1A = NULL, tcutd1B = NULL,
     hd1B = NULL, sd1B = NULL, tcutx0A = NULL, hx0A = NULL, sx0A = NULL,
     tcutx0B = NULL, hx0B = NULL, sx0B = NULL, tcutx1A = NULL,
     hx1A = NULL, sx1A = NULL, tcutx1B = NULL, hx1B = NULL, sx1B = NULL,
     noncompliance = c("none", "crossover", "mixed", "user"),
     gradual = FALSE, WtFun = c("FH", "SFH", "Ramp"), ppar = cbind(c(0, 0)),
     Spend.Info = c("Variance", "Events", "Hybrid(k)", "Calendar"), RR.Futility = NULL,
     qProp.one.or.Q = c("one", "Q"), Nsim = NULL, detail = FALSE, StatType = c("WLR",
     "ISD"), doProj=FALSE)
```

Arguments

EfficacyBoundary

This specifies the method used to construct the efficacy boundary. The available choices are:

'\((i)\) 'Lan-Demets(\(alpha=\text{total type I error}\), \(spending =\text{spending function}\)). The Lan-Demets method is based upon a error probability spending approach. The spending function can be set to \(ObrienFleming\), \(Pocock\), or \(Power(rho)\), where \(rho\) is the the power argument for the power spending function: \(rho=3\) is roughly equivalent to the O'Brien-Fleming spending function and smaller powers result in a less conservative spending function.

'\((ii)\) 'Haybittle(\(alpha=\text{total type I error}\), \(b.Haybittle=\text{user specified boundary point}\)). The Haybittle approach is conceptually the simplest of all methods for efficacy boundary construction. However, as it spends nearly no alpha until the end, is for all practical purposes equivalent to a single analysis design and to be considered overly conservative. This method sets all the boundary points equal to \(b.Haybittle\), a user specified value (try 3) for all analyses except the last, which is calculated so as to result in the total type I error, set with the argument alpha.

'\((iii)\) 'SC(\(be.end=\text{efficacy boundary point at trial end}\), \(prob=\text{threshold for conditional type I error for efficacy stopping}\)). The stochastic curtailment method is based upon the conditional probability of type I error given the current value of the statistic. Under this method, a sequence of boundary points on the standard normal scale (as are boundary points under all other methods) is calculated so that the total probability of type I error is maintained. This is done by considering the joint probabilities of continuing to the current analysis and then exceeding the threshold at the current analysis. A good value for the threshold value for the conditional type I error, \(prob\) is 0.90 or greater.

'\((iv)\) 'User supplied boundary points in the form \(c(b_1,b_2,b_3,\ldots,b_m)\), where \(m\) is the number of looks.

FutilityBoundary

This specifies the method used to construct the futility boundary. The available choices are:

'\((i)\) 'Lan-Demets(\(alpha=\text{total type II error}\), \(spending =\text{spending function}\)). The Lan-Demets method is based upon a error probability spending approach. The spending function can be set to \(ObrienFleming\), \(Pocock\), or \(Power(rho)\), where \(rho\) is the the power argument for the power spending function: \(rho=3\) is roughly equivalent to the O'Brien-Fleming spending function and smaller powers result in a less conservative spending function.

'\(\text{NOTE: there is no implementation of the Haybittle method for futility boundary construction. Given that the futility boundary depends upon values of the drift function, this method doesn't apply.}\)

'\((ii)\) 'SC(\(be.end=\text{efficacy boundary point at trial end}\), \(prob=\text{threshold for conditional type II error for futility stopping}\), \(drift.end=\text{projected drift at end of trial}\)). The stochastic curtailment method is based upon the conditional probability of type II error given the current value of the statistic. Under this method, a sequence of boundary points on the standard normal scale (as are boundary points under all other methods) is calculated so that the total proba-
bility of type II error, is maintained. This is done by considering the joint probabilities of continuing to the current analysis and then exceeding the threshold at the current analysis. A good value for the threshold value for the conditional type I error, prob is 0.90 or greater.

‘(iii) ’User supplied boundary points in the form c(b1, b2, b3, \ldots, b_m), where m is the number of looks.

NonBindingFutility
When using a futility boundary and this is set to 'TRUE', the efficacy boundary will be constructed in the absence of the futility boundary, and then the futility boundary will be constructed given the resulting efficacy boundary. This results in a more conservative efficacy boundary with true type I error less than the nominal level. This is recommended due to the fact that futility crossings are viewed by DSMB’s with much less gravity than an efficacy crossing and as such, the consensus is that efficacy bounds should not be discounted towards the null hypothesis because of paths which cross a futility boundary. Default value is 'TRUE'.

sided
Set to “2>” (quoted) for two sided tests of the null hypothesis when a positive drift corresponds to efficacy. Set to “2<” (quoted) for two sided tests of the null hypothesis when a negative drift corresponds to efficacy. Set to “1>” or “1<” for one sided tests of H0 when efficacy corresponds to a positive or negative drift, respectively. If method==“S” then this must be of the same length as StatType because the interpretation of sided is different depending upon whether StatType==”WLR” (negative is benefit) or StatType==”ISD” (positive is benefit)

method
Determines how to calculate the power. Set to ‘A’ (Asymptotic method, the default) or “S” (Simulation method)

accru
The upper endpoint of the accrual period beginning with time 0.

accrat
The rate of accrual per unit of time.

tlook
The times of planned interim analyses.

tcut0
Left hand endpoints for intervals upon which the arm-0 specific mortality is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity.

h0
A vector of the same length as tcut0 which specifies the piecewise constant arm-0 mortality rate.

s0
Alternatively, the arm-0 mortality distribution can be supplied via this argument, in terms of of the corresponding survival function values at the times given in the vector tcut0. If s0 is supplied, then h0is derived internally, assuming the piecewise exponential distribition. If you specify s0, the first element must be 1, and correspondingly, the first component of tcut0 will be the lower support point of the distribution. You must supply either h0 or s0 but not both.

tcut1
Left hand endpoints for intervals upon which the arm-1 specific mortality is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity.

rhaz
A vector of piecewise constant arm-1 versus arm-0 mortality rate ratios. If tcut1 and tcut0 are not identical, then tcut1, h0, and rhaz are internally rederived
at the union of the sequences \( t_{cut0} \) and \( t_{cut1} \). In all cases the arm-1 mortality rate is then derived at the time cutpoints \( t_{cut1} \) as \( rhaz \times h0 \).

\( h1 \)
Alternatively, the arm-1 mortality distribution can be supplied via this argument by specifying the piecewise constant arm-1 mortality rate. See the comments above.

\( s1 \)
Alternatively, the arm-1 mortality distribution can be supplied via this argument, in terms of of the corresponding survival function values at the times given in the vector \( t_{cut1} \). Comments regarding \( s0 \) above apply here as well. You must supply exactly one of the following: \( h1 \), \( rhaz \), or \( s1 \).

\( t_{cutc0} \)
Left hand endpoints for intervals upon which the arm-0 specific censoring distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity.

\( hc0 \)
A vector of the same length as \( t_{cutc0} \) which specifies the arm-0 censoring distribution in terms of a piecewise constant hazard function.

\( sc0 \)
Alternatively, the arm-0 censoring distribution can be supplied via this argument, in terms of of the corresponding survival function values at the times given in the vector \( t_{cutc0} \). See comments above. You must supply either \( hc0 \) or \( sc0 \) but not both.

\( t_{cutc1} \)
Left hand endpoints for intervals upon which the arm-1 specific censoring distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity.

\( hc1 \)
A vector of the same length as \( t_{cutc1} \) which specifies the arm-1 censoring distribution in terms of a piecewise constant hazard function.

\( sc1 \)
Alternatively, the arm-1 censoring distribution can be supplied via this argument, in terms of of the corresponding survival function values at the times given in the vector \( t_{cutc1} \). See comments above. You must supply either \( hc1 \) or \( sc1 \) but not both.

\( \text{noncompliance} \)
(i) Setting \( \text{noncompliance} \) to “none” for no non-compliance will automatically set the non-compliance arguments, below, to appropriate values for no compliance. This requires no additional user specification of non-compliance parameters. (ii) Setting \( \text{noncompliance} \) to “crossover” will automatically set crossover values in the arm 0/1 specific \( \text{post-cause-B-delay-mortality} \) for cross-over, i.e. simple interchange of the arm 0 and arm 1 mortalities. The user is required to specify all parameters corresponding to the arm 0/1 specific \( \text{cause-B-delay} \) distributions. The \( \text{cause-A-delay} \) and \( \text{post-cause-A-delay-mortality} \) are automatically set so as not to influence the calculations. Setting \( \text{noncompliance} \) to “mixed” will set the arm 0/1 specific \( \text{post-cause-B-delay-mortality} \) distributions for crossover as defined above. The user specifies the arm 0/1 specific \( \text{cause-B-delay} \) distribution as above, and in addition, all parameters related to the arm 0/1 specific \( \text{cause-A-delay} \) distributions and corresponding arm 0/1 specific \( \text{post-cause-A-delay-mortality} \) distributions. (iii) Setting \( \text{noncompliance} \) to “user” requires the user to specify all non-compliance distribution parameters.

\( t_{cutd0} \)
Left hand endpoints for intervals upon which the arm-0 specific \( \text{cause-A delay} \) distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Required only when \( \text{noncompliance} \) is set to “mixed” or “user”.

\( \text{noncompliance} \)
A vector of the same length as \( t_{\text{cutd0A}} \) containing piecewise constant hazard rates for the arm-0 cause-A delay distribution. Required only when noncompliance is set to "mixed" or "user".

When required, the arm-0 cause-A delay distribution is alternately specified via a survival function. A vector of the same length as \( t_{\text{cutd0A}} \).

Left hand endpoints for intervals upon which the arm-0 specific cause-B delay distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Always required when noncompliance is set to any value other than "none".

A vector of the same length as \( t_{\text{cutd0B}} \) containing piecewise constant hazard rates for the arm-0 cause-B delay distribution. Always required when noncompliance is set to any value other than "none".

When required, the arm-0 cause-B delay distribution is alternately specified via a survival function. A vector of the same length as \( t_{\text{cutd0B}} \).

Left hand endpoints for intervals upon which the arm-1 specific cause-A delay distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Required only when noncompliance is set to "mixed" or "user".

A vector of the same length as \( t_{\text{cutd1A}} \) containing piecewise constant hazard rates for the arm-1 cause-A delay distribution. Required only when noncompliance is set to "mixed" or "user".

When required, the arm-1 cause-A delay distribution is alternately specified via a survival function. A vector of the same length as \( t_{\text{cutd1A}} \).

Left hand endpoints for intervals upon which the arm-1 specific cause-B delay distribution hazard function is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Always required when noncompliance is set to any value other than "none".

A vector of the same length as \( t_{\text{cutd1B}} \) containing piecewise constant hazard rates for the arm-1 cause-B delay distribution. Always required when noncompliance is set to any value other than "none".

When required, the arm-1 cause-A delay distribution is alternately specified via a survival function. A vector of the same length as \( t_{\text{cutd1B}} \).

Left hand endpoints for intervals upon which the arm-0 specific post-cause-A-delay-mortality rate is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Required only when noncompliance is set to "mixed" or "user".

A vector of the same length as \( t_{\text{cutx0A}} \) containing the arm-0 post-cause-A-delay mortality rates. Required only when noncompliance is set to "mixed" or "user".

When required, the arm-0 post-cause-A-delay mortality distribution is alternately specified via a survival function. A vector of the same length as \( t_{\text{cutx0A}} \).

Left hand endpoints for intervals upon which the arm-0 specific post-cause-B-delay-mortality rate is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Always required when noncompliance is set to any value other than "none".
hx0B A vector of the same length as tcux0B containing the arm-0 post-cause-B-delay mortality rates. Always required when noncompliance is set to any value other than "none".

sx0B When required, the arm-0 post-cause-B-delay mortality distribution is alternately specified via a survival function. A vector of the same length as tcux0B.

tcux1A Left hand endpoints for intervals upon which the arm-1 specific post-cause-A-delay mortality rate is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Required only when noncompliance is set to "mixed" or "user".

hx1A A vector of the same length as tcux1A containing the arm-1 post-cause-A-delay mortality rates. Required only when noncompliance is set to "mixed" or "user".

sx1A When required, the arm-1 post-cause-A-delay mortality distribution is alternately specified via a survival function. A vector of the same length as tcux1A.

tcux1B Left hand endpoints for intervals upon which the arm-1 specific post-cause-B-delay mortality rate is constant. The last given component is the left hand endpoint of the interval having right hand endpoint infinity. Always required when noncompliance is set to any value other than "none".

hx1B A vector of the same length as tcux1B containing the arm-1 post-cause-B-delay mortality rates. Always required when noncompliance is set to any value other than "none".

sx1B When required, the arm-1 post-cause-B-delay mortality distribution is alternately specified via a survival function. A vector of the same length as tcux1B.

gradual Should the conversion to post-noncompliance mortality be gradual. Under the default behavior, gradual=FALSE, there is an immediate conversion to the post-noncompliance mortality rate function. If gradual is set to TRUE then this conversion is done "gradually". In truth, at the individual level what is done is that the new mortality rate function is a convex combination of the pre-noncompliance mortality and the post-noncompliance mortality, with the weighting in proportion to the time spent in compliance with the study arm protocol.

WtFun Specifies the name of a weighting function (of time) for assigning relative weights to events according to the times at which they occur. The default, "FH", a two parameter weight function, specifies the ‘Fleming-Harrington’ g-rho family of weighting functions defined as the pooled arm survival function (Kaplan-Meier estimate) raised to the g times its complement raised to the rho. Note that g=rho=0 corresponds to the unweighted log-rank statistic. A second choice is the “SFH” function, (for ‘Stopped Fleming-Harrington’), meaning that the “FH” weights are capped at their value at a user specified time, which has a total of 3 parameters. A third choice is Ramp(tcut). Under this choice, weights are assigned in a linearly manner from time 0 until a user specified cut-off time, tcut, after which events are weighted equally. It is possible to conduct computations on nstat candidate statistics within a single run. In this case, WtFun should be a character vector of length nstat having components set from among the available choices.

ppar A vector containing all the weight function parameters, in the order determined by that of “WtFun”. For example, if WtFun is set to c("FH", "SFH", "Ramp")
then ppar should be a vector of length six, with the “FH” parameters in the first two elements, “SFH” parameters in the next 3 elements, and “Ramp” parameter in the last element.

**RR.Futility**  
The relative risk corresponding to the alternative hypothesis that is required in the construction of the futility boundary. Required if Boundary.Futility is set to a non-null value.

**Spend.Info**  
When the test statistic is something other than the unweighted log-rank statistic, the variance information, i.e. the ratio of variance at interim analysis to variance at the end of trial, is something other than the ratio of events at interim analysis to the events at trial end. The problem is that in practice one doesn’t necessarily have a good idea what the end of trial variance should be. In this case the user may wish to spend the type I and type II error probabilities according to a different time scale. Possible choices are “Variance”, (default), which just uses the variance ratio scale, “Events”, which uses the events ratio scale, “Hybrid(k)”, which makes a linear transition from the “Variance” scale to the “Events” scale beginning with analysis number k. The last choice, “Calendar”, uses the calendar time scale.

**qProp.one.or.Q**  
If a futility boundary is specified, what assumption should be made about the drift function (the mean value of the weighted log-rank statistic at analysis j normalized by the square root of the variance function at analysis k). In practice we don’t presume to know the shape of the drift function. Set to “one” or “Q”. The choice “one” results in a more conservative boundary.

**Nsim**  
If you specify method==“S”, then you must specify the number of simulations. 1000 should be sufficient.

**detail**  
If you specify method==“S”, and want to see the full level of detail regarding arguments returned from the C level code, specify detail==TRUE

**StatType**  
If you specify method==“S”, then the available choices are “WLR” (weighted log-rank) and “ISD” (integrated survival difference).

**doProj**  
Works only when method==“S”. If a weighted log-rank statistic is specified without maximum information having been stipulated in the design then certain functionals, the Q first and second moments, must be projected. Setting this argument to TRUE includes this projection into the simulation runs.

**Value**

Returns a value of class PwrGSD which has components listed below. Note that the print method will display a summary table of estimated powers and type I errors as a nstat by 2 matrix. The summary method returns the same object invisibly, but after computing the summary table mentioned above, and it is included in the returned value as a component TBL. See examples below.

**dPower**  
A length(tlook) by nstat matrix containing in each column, an increment in power that resulted at that analysis time for the given statistic.

**dErrorI**  
A length(tlook) by nstat matrix containing in each column, an increment in type I error that resulted at that analysis time for the given statistic. Always sums to the total alpha specified in alphatot.
A list with components equal to the arguments of the C-call, which correspond in a natural way to the arguments specified in the R call, along with the computed results in `palphavec`, `palphavec`, `inffrac`, and `mu`. The first two are identical to `dErrorI` and `dPower`, explained above. The last two are `length(tlook)` by `nstat` matrices. For each statistic specified in `par`, the corresponding columns of `inffrac` and `mu` contain the information fraction and drift at each of the analysis times.

Author(s)
Grant Izmirlian <izmirlian@nih.gov>

References
Izmirlian G. (2014). Estimation of the relative risk following group sequential procedure based upon the weighted log-rank statistic. Statistics and its Interface 7(1), 27-42

See Also
cpd.PwrGSD

Examples

```r
library(PwrGSD)

test.example <-
PwrGSD(EfficacyBoundary = LanDemets(alpha = 0.05, spending = ObrienFleming),
FutilityBoundary = LanDemets(alpha = 0.1, spending = ObrienFleming),
RR.Futility = 0.82, sided="<", method="A", accrat =9818.65,
tlook =c(7.14, 8.14, 9.14, 10.14, 10.64, 11.15, 12.14, 13.14,
tcut0 =0:19, h0 =c(rep(3.73e-04, 2), rep(7.45e-04, 3),
rep(1.49e-03, 15)),
tcut1 =0:19, rhaz =c(1, 0.9125, 0.8688, 0.7814, 0.6941,
0.6943, 0.6072, 0.5202, 0.4332, 0.3517, 0.2702,
0.6524, 0.6527, 0.6530, 0.6534, 0.6537,
0.6541, 0.6544, 0.6547, 0.6551, 0.6554),
tcutc0 =0:19, hc0 =c(rep(1.05e-02, 2), rep(2.09e-02, 3),
rep(1.49e-03, 15)),
tcutc1 =0:19, hc1 =c(rep(1.05e-02, 2), rep(2.09e-02, 3),
```
Relative cumulative mortality to Relative Risk

Description

Given the relative cumulative mortality (ratio of CDFs), the baseline hazard and censoring hazard at a grid of time points, calculates the corresponding risk ratio at a second specified grid of time points.

Usage

RCM2RR(tlook, tcut.i, h.i, h0th, accru, rcm)

Arguments

tlook  Second grid of time points at which you desire risk ratios
tcut.i  First grid of time points at which baseline hazard, censoring hazard and relative cumulative mortality are specified (left hand endpoints of intervals)
h.i  Values of baseline hazard on intervals given by tcut.i
h0th  Values of censoring hazard on intervals given by tcut.i
accru  Time at which uniform accrual is completed (starts at 0)
rcm  Values of relative cumulative mortality (ratio of CDFs) on intervals given by tcut.i

Value

Values of risk ratio on intervals given by tlook

Author(s)

Grant Izmirlian <izmirlian@nih.gov>
**RR2RCM**

*Relative risk to Relative Cumulative Mortality*

**Description**

Relative risk to Relative Cumulative Mortality

**Usage**

```
RR2RCM(tlook, tcut.i, tcut.ii, h, rr, h0th, accru)
```

**Arguments**

- `tlook`: Grid of time points at which you desire cumulative relative mortality
- `tcut.i`: Grid of time points at which baseline hazard, censoring hazard and relative cumulative mortality are specified (left hand endpoints of intervals)
- `tcut.ii`: Grid of time points at which study arm hazard is specified (left hand endpoints of intervals)
- `h`: Values of baseline hazard on intervals given by `tcut.i`
- `rr`: Values of risk ratio on intervals given by `tcut.i`
- `h0th`: Values of censoring hazard on intervals given by `tcut.i`
- `accru`: Time at which uniform accrual is completed (starts at 0)

**Value**

Values of relative cumulative mortality (ratio of CDFs) on intervals given by `tlook`

**Author(s)**

Grant Izmirlian <izmirlian@nih.gov>

---

**SC**

*The Stochastic Curtailment method of Boundary Construction*

**Description**

The function `SC` is used in calls to the functions `GrpSeqBnds` and `PwrGSD` as a possible setting for the arguments `EfficacyBoundary` and `FutilityBoundary`, in specification of the method whereby efficacy and or futility boundaries are to be constructed. The Stochastic Curtailment method is one of four currently available choices, the others being LanDemets, Haybittle (efficacy only) and user specified.

**Usage**

```
SC(be.end, prob, drift.end = NULL, from = NULL, to = NULL)
```
Arguments

be.end  The value of the efficacy criterion in the scale of a standardized normal. This should be set to something further from the null than the single test $Z_{\alpha}$. For example if the total type I error probability is 0.05 in a two-sided test of the null than set be.end to 2.10 or larger (instead of 1.96).

prob    The criterion, a probability to be exceeded in order to stop. 0.90 or above is a good choice. See detail below.

drift.end  Required only if you are using SC to set the FutilityBoundary. In this case, set drift.end to the value of the drift function anticipated at the end of the trial. See detail below.

from

Warning EXPERIMENTAL: you can actually construct boundaries via a hybrid of the 3 boundary construction methods, LanDemets, SC, and 'user specified'. When using a hybrid boundary, set the argument EfficacyBoundary or FutilityBoundary respectively, to a list with components LanDemets, SC, or user specified numbers. In the former two cases, from and to are used in LanDemets and also in SC to stipulate how many interim analyses they are in effect. See the help for GrpSeqBnds and PwrGSD

to

See above.

Details

When the stochastic curtailment procedure is used to construct the efficacy boundary, i.e. EfficacyBoundary=SC(...), the efficacy criterion is reached when the conditional probability, under the null hypothesis, that the last analysis results in statistical significance, given the present value of the statistic, exceeds 'prob'. In of itself, this doesn't produce a boundary on the scale of a standard normal, but it is easily converted to one as is done here. When this is used to construct a futility boundary, i.e. FutilityBoundary=SC(...), the futility criterion is reached when the conditional probability, under the design alternative hypothesis, that the last analysis does not result in statistical significance, given the present value of the statistic, exceeds 'prob'. The design alternative corresponds to a drift function, which is the expected value of the statistic normalized to have variance equal to the information fraction at each interim analysis. For the unweighted log-rank statistic, the drift function is $(V_T)^{(1/2)} B f$, where B is the logged relative risk, $V_T$ is the variance at the end of the trial and f is the information fraction. If the two trial arms are balanced and the number at risk is roughly constant throughout the trial then $V_T = \pi (1-\pi) N_T$, where $\pi$ is the constant proportion at risk in one of the trial arms and $N_T$ is the anticipated number of events.

Value

An object of class boundary_construction_method which is really a list with the following components. The print method displays the original call.

type    Gives the boundary construction method type, which is the character string "SC"

be.end  The numeric value passed to the argument 'be.end', which is the value of the efficacy criterion in the scale of a standardized normal.

prob    The numeric value passed to the argument 'prob', which is the probability to be exceeded in order to stop.
drift.end  The numeric value passed to the argument 'drift.end', which is the value of the
  drift function at the end of the trial. See details.
from  The numeric value passed to the argument 'from'. See above.
to  The numeric value passed to the argument 'to'. See above.
call  returns the call

Note
The print method returns the call by default

Author(s)
Grant Izmirlian

References
see references under PwrGSD

See Also
LanDemets, GrpSeqBnds, PwrGSD

Examples
```r
## example 1: what is the result of calling a Boundary Construction Method function
## A call to 'SC' just returns the call
SC(be.end=2.10, prob=0.90)

## It does argument checking...this results in an error
## Not run:
SC(be.end=2.10)

## End(Not run)

## but really its value is a list with the a component containing
## the boundary method type, "LanDemets", and components for each
## of the arguments.
names(SC(be.end=2.10, prob=0.90))

SC(be.end=2.10, prob=0.90, drift.end=2.34)$type
SC(be.end=2.10, prob=0.90, drift.end=2.34)$be.end
SC(be.end=2.10, prob=0.90, drift.end=2.34)$prob
SC(be.end=2.10, prob=0.90, drift.end=2.34)$drift.end

## example 2: ...But the intended purpose of the spending functions is
## in constructing calls to 'GrpSeqBnds' and to 'PwrGSD':

frac <- c(0.07614902, 0.1135391, 0.168252, 0.2336901, 0.3186155,
  0.4164776, 0.5352199, 0.670739, 0.8246061, 1)
drift <- c(0.3836636, 0.5117394, 0.6918584, 0.8657705, 1.091984,
```
SCtoBdry

1.311094, 1.538582, 1.818346, 2.081775, 2.345386

test <- GrpSeqBnds(frac=frac, EfficacyBoundary=LanDemets(alpha=0.05, spending=ObrienFleming),
                   FutilityBoundary=SC(be.end=2.10, prob=0.90, drift.end=drift[10]),
                   drift=drift)

SCtoBdry

Converts a stochastic curtailment boundary (conditional type I or II error probability) into a (efficacy or futility) boundary on the standardized Z scale

Description

Converts a stochastic curtailment boundary (conditional type I or II error probability) into a (efficacy or futility) boundary on the standardized Z scale

Usage

SCtoBdry(prob, frac, be.end, drift = NULL, drift.end = NULL)

Arguments

prob

The stochastic curtailment threshold probability, which is the complement of the type I (efficacy) or II (futility) error. We typically use 0.90 which will stop for efficacy if the probability under the null that the final analysis results in an efficacious decision given the data so far exceeds 0.90, and stops for futility of the probability under the alternative corresponding to the drift arguments, that the final analysis results in a futility decision given the data so far, exceeds 0.90.

frac

The variance ratio. See the GrpSeqBnds documentation for details.

be.end

Value of efficacy (futility) boundary at the final analysis

drift

The drift function. See the GrpSeqBnds documentation for details.

drift.end

Required if using a futility boundary. This is the value of the drift function at the final analysis. Must be projected using the trial design.

Value

A efficacy or futility boundary on the standard normal scale

Author(s)

Grant Izmirlian
SimGSB

Verifies the results of "GrpSeqBnds" via simulation

Description

Verifies the results of GrpSeqBnds via simulation

Usage

SimGSB(object, nsim = 1e+05, ...)

Arguments

object an object of class either boundaries or PwrGSD
nsim number of simulations to do
... if object is of class PwrGSD and there are more than one statistic under investigation, then you may specify an argument stat. The default value is 1, meaning the first one.
**wtdlogrank**

**Value**
A tabulation of the results

**Author(s)**
Grant Izmirlian <izmirlian@nih.gov>

**See Also**
GrpSeqBnds

**Examples**

```r
## none as yet
```

---

**wtdlogrank**

*Weighted log-rank test*

**Description**
Computes a two sample weighted log-rank statistic with events weighted according to one of the available weighting function choices.

**Usage**

```r
wtdlogrank(formula = formula(data), data = parent.frame(), WtFun = c("FH", "SFH", "Ramp"),
            param = c(0, 0), sided = c(2, 1), subset, na.action, w = FALSE)
```

**Arguments**

- `formula`:
  A formula of the form `Surv(Time,Event) ~ arm` where `arm` is a dichotomous variable with values 0 and 1.

- `data`:
  A dataframe.

- `WtFun`:
  A selection from the available list: “FH” (Fleming-Harrington), “SFH” (stopped Fleming-Harrington) or “Ramp”. See `param` in the following line.

- `param`:
  Weight function parameters. Length and interpretation depends upon the selected value of `WtFun`:
  - If `WtFun` equals “FH” then `param` is a length 2 vector specifying the power of the pooled (across arms) kaplan meier estimate and its complement.
  - If `WtFun` equals “SFH” then `param` is a length 3 vector with first two components as in the “FH” case, and third component the time (in the same units as the time to event) at which the “FH” weight function is capped off at its current value.
  - If `WtFun` equals “Ramp” then `param` is of length 1 specifying the time (same units as time to event) at which events begin to get equal weight. The “Ramp” weight function is a linearly increasing deterministic weight function which is capped off at 1 at the user specified time.
sided One or Two sided test? Set to 1 or 2
subset Analysis can be applied to a subset of the dataframe based upon a logical ex-
pression in its variables
na.action Method for handling NA values in the covariate, arm
w currently no effect

Value

An object of class survtest containing components

pn sample size
wtyp internal representation of the WtFun argument
par internal representation of the param argument
time unique times of events across all arms
nrisk number at risk across all arms at each event time
nrisk1 Number at risk in the experimental arm at each event time
nevent Number of events across all arms at each event time
nevent1 Number of events in the experimental arm at each event time
wt Values of the weight function at each event time
pntimes Number of event times
stat The un-normalized weighted log-rank statistic, i.e. the summed weighted ob-
erved minus expected differences at each event time
var Variance estimate for the above
UQt Cumulative sum of increments in the sum resulting in stat above
varQt Cumulative sum of increments in the sum resulting in var above
var1t Cumulative sum of increments in the sum resulting in the variance of an un-
weighted version of the statistic
pu0 person units of follow-up time in the control arm
pu1 person units of follow-up time in the intervention arm
n0 events in the control arm
n1 events in the intervention arm
n sample size, same as pn
call the call that created the object

Author(s)

Grant Izmirlian <izmirlian@nih.gov>

References

data. Biometrika 69, 553-566.
wtdlogrank

See Also

IntSurvDiff

Examples

```r
library(PwrGSD)
data(lung)
fit.wlr <- wtdlogrank(Surv(time, I(status==2)) ~ I(sex==2), data=lung, WtFun="SFH", param=c(0,1,300))
```
Index

* Bias Adjusted point estimate
  EX1gXK, 14
* Sequential Design
  EX1gXK, 14
* Stage-wise P-value
  gsd.dens, 19
* datasets
  lung, 28
* data
  cpd.PwrGSD, 5
  Elements, 13
  mystack, 29
  Power, 36
* design
  as.boundaries, 3
  CondPower, 4
  GrpSeqBnds, 15
  Haybittle, 21
  LanDemets, 25
  ObrienFleming, 31
  Pocock, 33
  Pow, 34
  PwrGSD, 37
  SC, 46
  SCToBdry, 49
* hplot
  plot.cpd.PwrGSD, 32
* htest
  as.boundaries, 3
  CondPower, 4
  GrpSeqBnds, 15
  Haybittle, 21
  LanDemets, 25
  ObrienFleming, 31
  Pocock, 33
  Pow, 34
  PwrGSD, 37
  SC, 46
  SCToBdry, 49
* manip
  agghaz, 2
* misc
  DX, 12
  lookup, 27
  SimGSB, 50
* sequential statistic pdf
  gsd.dens, 19
* survival
  agghaz, 2
  CDFOR2LRR, 4
  CRRtoRR, 10
  CY2TOShaz, 11
  IntSurvDiff, 23
  mysurvfit, 30
  plot.cpd.PwrGSD, 32
  PwrGSD, 37
  RCM2RR, 45
  RR2RCM, 46
  wtdlogrank, 51

agghaz, 2
as.boundaries, 3
CDFOR2LRR, 4
CondPower, 4
cpd.PwrGSD, 5, 13, 33, 37, 44
CRRtoRR, 10
CY2TOShaz, 11
detail, 12
DX, 12
Elements, 6, 13, 33
EX1gXK, 14, 21
GrpSeqBnds, 3, 5, 15, 22, 26, 31, 34, 35, 48, 49, 51
gsd.dens, 14, 19
Haybittle, 21
INDEX

IntSurvDiff, 23, 53
Landemets, 22, 25, 31, 34, 35, 48
lookup, 27
lung, 28
mystack, 29
mysurvf, 30
ObrienFleming, 26, 31, 34, 35
plot.cpd.PwrGSD, 6, 32
Pocock, 26, 31, 33, 35
Pow, 26, 31, 34, 34
Power, 6, 33, 36
PwrGSD, 13, 18, 22, 26, 31, 33–35, 37, 37, 48
RCM2RR, 45
RR2RCM, 46
SC, 22, 26, 46
SCtoBdry, 49
SimGSB, 50
wtdlogrank, 25, 51