

# Package ‘clinfun’

November 22, 2009

**Title** Clinical Trial Design and Data Analysis Functions

**Version** 0.8.7

**Depends** R (>= 2.0.0), graphics, stats

**Imports** mvtnorm

**Suggests** survival

**Author** Venkatraman E. Seshan

**Description** Utilities to make your clinical collaborations easier if not fun.

**Maintainer** Venkatraman E. Seshan <seshanv@mskcc.org>

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-11-22 16:11:35

## R topics documented:

calogrank . . . . .	2
coxphCPE . . . . .	3
coxphQuantile . . . . .	3
fedesign . . . . .	4
gsdesign . . . . .	6
oc.twostage.bdry . . . . .	7
permtests . . . . .	8
ph2simon . . . . .	8
ph2single . . . . .	9
power.ladesign . . . . .	10
pselect . . . . .	11
toxbdry . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

`calogrank`*Survival curves analysis of covariance*

---

## Description

Logrank test to compare survival curves adjusting for covariates

## Usage

```
calogrank(ftime, fstatus, grp, cvt, strat=NULL)
```

## Arguments

<code>ftime</code>	failure times
<code>fstatus</code>	status indicator
<code>grp</code>	group indicator
<code>cvt</code>	continuous covariates used for adjusted analysis
<code>strat</code>	stratification variable

## Details

`calogrank` is the covariate adjusted version of k-sample `survdiff`. The function in its current form only does basic error checking.

## References

Heller G. and Venkatraman E.S. (2004) A nonparametric test to compare survival distributions with covariate adjustment. *JRSS-B* 66, 719-733.

## Examples

```
## Not run:  library(survival)
data(pbc)
pbc1 <- pbc
pbc1$trt[pbc1$trt == -9] <- NA
pbc1$copper[pbc1$copper == -9] <- NA
calogrank(pbc1$time, pbc1$status, pbc1$trt, pbc1[,c("copper")])
calogrank(pbc1$time, pbc1$status, pbc1$trt, pbc1[,c("protime", "copper")])
## End(Not run)
```

---

`coxphCPE`*Gonen & Heller Concordance Probability Estimate*

---

**Description**

Calculates the Concordance Probability Estimate for a Cox model.

**Usage**

```
coxphCPE(phfit)
```

**Arguments**

`phfit` output from a proportional hazards fit.

**Value**

`coxphCPE` returns a vector with `CPE`, `smooth.CPE` & `se.CPE` which are the estimate, the smoothed estimate and its standard error respectively.

**References**

Gonen M and Heller G. (2005) Concordance probability and discriminatory power in proportional hazards regression. *Biometrika* 92, 965-970.

**Examples**

```
## Not run: library(survival)
data(pbc)
pbcfit <- coxph(Surv(time, status) ~ trt + log(copper), pbc,
  subset=(trt>0 & copper>0))
coxphCPE(pbcfit)

## End(Not run)
```

---

`coxphQuantile`*Survival time quantile as a function of covariate*

---

**Description**

Draws a quantile curve of survival distribution as a function of covariate.

**Usage**

```
coxphQuantile(phfit, xrange, p=0.5, whichx=1, otherx=NULL, ...)
```

**Arguments**

phfit	output from a proportional hazards fit.
xrange	the range of covariate values for which the quantiles of survival times are computed.
p	the probability level for the quantile (default is median).
whichx	if there are more than one covariates in the Cox model, the one chosen for the quantile plot.
otherx	the values for other covariates in the Cox model. If missing uses their average values.
...	additional parameters to be passed on to the lines command.

**Details**

This function is used to draw quantile curves. It requires a plot of the data (time & covariate of interest) to be present. See example.

It invisibly returns the observed failure times and the covariate values at which the estimated survival probability is (exactly) p.

**References**

Heller G. and Simonoff J.S. (1992) Prediction in censored survival data: A comparison of the proportional hazards and linear regression models. *Biometrics* 48, 101-115.

**Examples**

```
## Not run:  library(survival)
data(pbc)
pbcfit <- coxph(Surv(time, status) ~ trt + log(copper), pbc,
  subset=(trt>0 & copper>0))
plot(log(pbc$copper[pbc$trt>0 & pbc$copper>0]), pbc$time[pbc$trt>0 &
  pbc$copper>0], pch=c("o","x")[1+pbc$status[pbc$trt>0 & pbc$copper>0]],
  xlab="log Copper", ylab="Survival time")
coxphQuantile(pbcfit, c(2.5,6), whichx=2, otherx=1)
coxphQuantile(pbcfit, c(2.5,6), p=0.75, whichx=2, otherx=2, col=2)
## End(Not run)
```

**Description**

Calculates sample size, effect size and power based on Fisher's exact test

**Usage**

```

fe.ssize(p1, p2, alpha=0.05, power=0.8, r=1, npm=5, mmax=1000)
CPS.ssize(p1, p2, alpha=0.05, power=0.8, r=1)
fe.mdor(ncase, ncontrol, pcontrol, alpha=0.05, power=0.8)
fe.power(d, n1, n2, p1, alpha = 0.05)

```

**Arguments**

p1	response rate of standard treatment
p2	response rate of experimental treatment
d	difference = p2-p1
pcontrol	control group probability
n1	sample size for the standard treatment group
n2	sample size for the standard treatment group
ncontrol	control group sample size
ncase	case group sample size
alpha	size of the test (default 5%)
power	power of the test (default 80%)
r	treatments are randomized in 1:r ratio (default r=1)
npm	the sample size program searches for sample sizes in a range (+/- npm) to get the exact power
mmax	the maximum group size for which exact power is calculated

**Details**

CPS.ssize returns Casagrande, Pike, Smith sample size which is a very close to the exact. Use this for small differences p2-p1 (hence large sample sizes) to get the result instantaneously.

fe.ssize return a 2x3 matrix with CPS and Fisher's exact sample sizes with power.

fe.mdor return a 3x2 matrix with Schlesselman, CPS and Fisher's exact minimum detectable odds ratios and the corresponding power.

fe.power returns a Kx2 matrix with probabilities (p2) and exact power.

**References**

Casagrande, JT., Pike, MC. and Smith PG. (1978). An improved approximate formula for calculating sample sizes for comparing two binomial distributions. *Biometrics* 34, 483-486.

Fleiss, J. (1981) Statistical Methods for Rates and Proportions.

Schlesselman, J. (1987) Re: Smallest Detectable Relative Risk With Multiple Controls Per Case. *Am. J. Epi.*

**Description**

Functions to calculate sample size for group sequential designs

**Usage**

```
gsdesign.binomial(frac, pC, pE, sig.level = 0.05, power = 0.8,
  delta.eb=0.5, delta.fb = 0, alternative = c("two.sided", "one.sided"),
  tol=0.0001)
gsdesign.normal(frac, delta, sd = 1, sig.level = 0.05, power = 0.8,
  delta.eb = 0.5, delta.fb = 0, alternative = c("two.sided", "one.sided"),
  tol=0.0001)
gsdesign.survival(frac, haz.ratio, sig.level = 0.05, power = 0.8,
  delta.eb = 0.5, delta.fb = 0, alternative = c("two.sided", "one.sided"),
  tol=0.0001)
```

**Arguments**

<code>frac</code>	information fraction or the ratio of current sample size or number of events to the total sample size or number of events. This should be an increasing vector of numbers from 0 to 1 with the last one being 1. If just 1 is given a fixed sample design is derived.
<code>pC</code>	prob of success of the standard therapy (for binomial data)
<code>pE</code>	prob of success of the experimental therapy (for binomial data)
<code>delta</code>	true difference in means (for normal data)
<code>sd</code>	standard deviation (for normal data)
<code>haz.ratio</code>	hazard ratio (for survival comparison)
<code>sig.level</code>	significance level (type I error probability)
<code>power</code>	power of test (1 minus type II error probability)
<code>delta.eb</code>	power for efficacy boundary in the Pocock (power=0) to O'Brien-Fleming (power=0.5) family (default is 0.5)
<code>delta.fb</code>	power for futility boundary in the Pocock (power=0) to O'Brien-Fleming (power=0.5) family (default is 0.5)
<code>alternative</code>	one- or two-sided test.
<code>tol</code>	tolerance level for multivariate normal probability computation.

**Value**

a list with ifrac, sig.level, power, alternative, delta.eb and:

efbdry	the critical value to use at the different looks.
sample.size	the sample size per arm for binomial/normal data.
num.events	the total number of failures which should be converted to number of subjects using censoring proportion.

---

oc.twostage.bdry *Two-stage boundary operating characteristics*

---

**Description**

Calculates the operating characteristics of a two-stage boundary.

**Usage**

```
oc.twostage.bdry(pu, pa, r1, n1, r, n)
```

**Arguments**

pu	unacceptable response rate
pa	response rate that is desirable
r1	first stage threshold to declare treatment undesirable
n1	first stage sample size
r	overall threshold to declare treatment undesirable
n	total sample size

**Value**

oc.twostage.bdry returns the type I and II error rates as well as the probability of early termination and expected sample size under pu for a specific boundary.

---

 permtests

*Permutation versions of some common tests*


---

**Description**

Small sample tests using the permutation reference distributions.

**Usage**

```
permlogrank(formula, data, subset, na.action, rho=0, nperm=5000)
jonckheere.test(x, g, alternative = c("two.sided", "increasing",
                                     "decreasing"))
```

**Arguments**

nperm            number of permutations for the reference distribution  
 formula, data, subset, na.action, rho  
                   see survdiff for details  
 x, g             data and group vector  
 alternative     means are monotonic (two.sided), increasing, or decreasing

**Details**

permlogrank is the permutation version of k-sample survdiff

jonckheere.test is the exact (permutation) version of the Jonckheere-Terpstra test. It uses the statistic

$$\sum_{k < l} \sum_{ij} I(X_{ik} < X_{jl}) + 0.5I(X_{ik} = X_{jl}),$$

where  $i, j$  are observations in groups  $k$  and  $l$  respectively. The asymptotic version is equivalent to `cor.test(x, g, method="k")`.

---

 ph2simon

*Simon's 2-stage Phase II design*


---

**Description**

Calculates Optimal and Minimax 2-stage Phase II designs given by Richard Simon

**Usage**

```
ph2simon(pu, pa, ep1, ep2, nmax=100)
## S3 method for class 'ph2simon':
print(x, ...)
## S3 method for class 'ph2simon':
plot(x, ...)
```

**Arguments**

pu	unacceptable response rate
pa	response rate that is desirable
ep1	threshold for the probability of declaring drug desirable under p0
ep2	threshold for the probability of rejecting the drug under p1
nmax	maximum total sample size (default 100; can be at most 500)
x	object returned by ph2simon
...	arguments to be passed onto plot and print commands called within

**Value**

ph2simon returns a list with pu, pa, alpha, beta and nmax as above and:

out	matrix of best 2 stage designs for each value of total sample size n. the 6 columns are: r1, n1, r, n, EN(p0), PET(p0)
-----	--

The "print" method formats and returns the minimax and optimal designs. The "plot" plots the expected sample size against the maximum sample size as in Jung et al., 2001

**References**

- Simon R. (1989). Optimal Two-Stage Designs for Phase II Clinical Trials. *Controlled Clinical Trials* 10, 1-10.
- Jung SH, Carey M and Kim KM. (2001). Graphical Search for Two-Stage Designs for Phase II Clinical Trials. *Controlled Clinical Trials* 22, 367-372.

**Examples**

```
ph2simon(0.2, 0.4, 0.1, 0.1)
ph2simon(0.2, 0.35, 0.05, 0.05)
ph2simon(0.2, 0.35, 0.05, 0.05, nmax=150)
```

---

ph2single

*Exact single stage Phase II design*

---

**Description**

Calculates the exact one stage Phase II design

**Usage**

```
ph2single(pu, pa, ep1, ep2, nsoln=5)
```

**Arguments**

pu	unacceptable response rate
pa	response rate that is desirable
ep1	threshold for the probability of declaring drug desirable under p0
ep2	threshold for the probability of rejecting the drug under p1
nsoln	number of designs with given alpha and beta

**Value**

ph2single returns a data frame with variables: n, r, and the Type I and Type II errors.

---

power.ladesign	<i>Power of k-sample rank test under Lehmann alternative</i>
----------------	--

---

**Description**

Functions to calculate the power of rank tests for animal studies.

**Usage**

```
power.ladesign(gsize, odds.ratio, sig.level = 0.05, statistic =
  c("Kruskal-Wallis", "Jonckheere-Terpstra"), alternative =
  c("two.sided", "one.sided"), nrep=1e+6)
## S3 method for class 'ladesign':
print(x, ...)
```

**Arguments**

gsize	sample size of the k (= length of vector) groups.
odds.ratio	odds ratio parameters for the k-1 groups. The first group is considered the control.
sig.level	the significance level of the test (default = 0.05)
statistic	the test statistic for the k-group comparison. Is one of Kruskal-Wallis (default) or Jonckheere-Terpstra.
alternative	one- or two-sided test. Valid only for the Jonckheere-Terpstra test.
nrep	number of reps (default 1 million) for Monte Carlo.
x	object of class ladesign returned by power.ladesign
...	arguments to be passed on left for S3 method consistency.

**Details**

Although the power for Jonckheere-Terpstra test is calculated for any set of odds ratio, the test is meant for monotone alternative. Thus it is preferable to specify odds ratios that are monotonically increasing with all values larger than 1 or decreasing with all values smaller than 1.

**Value**

returns a list with objects group.size, odds.ratio, statistic, sig.level and power. The "print" method formats the output.

**References**

Heller G. (2006). Power calculations for preclinical studies using a K-sample rank test and the Lehmann alternative hypothesis. *Statistics in Medicine* 25, 2543-2553.

**Examples**

```
power.ladesign(c(9,7), 4, statistic="K")
power.ladesign(c(9,7,9), c(2,4), statistic="J")
power.ladesign(c(9,7,9), c(2,4), statistic="J", alt="o")
```

---

pselect

*Probability of selection under pick the winner rule*

---

**Description**

Calculates the probability of selecting the treatment with the higher response rate under the pick the winner rule.

**Usage**

```
pselect(n, p, min.diff=NULL, min.resp=NULL)
```

**Arguments**

n	sample size for each treatment arm. This is either a single integer or a vector of two integers for the special case of comparing two treatments with unequal sample sizes
p	vector of response rates for the treatments.
min.diff	this is the number of responses or the rate by which the best treatment should be superior to the others to be chosen. This must be a positive integer or a rate between 0 and 1. If missing it defaults to 1 for the equal sample size case but quits with a warning for the unequal sample size case.
min.resp	the minimum number of responses in each treatment arm for it to be considered further. If missing defaults to 0.

**Value**

the function returns a list with:

`prob.none.worthy`

is the probability that no treatment has the minimum number of responses specified in `min.resp`. this element is present only if `min.resp` is greater than 0 for at least one arm.

`prob.inconclusive`

this is the probability that the best treatment has the requisite `min.resp` responses but exceeds the second best by less than `min.diff` responses (rate) provided the second best also has at least `min.resp` responses.

`prob.selection`

this is a matrix which for each treatment gives the response probability and the probability of selecting it i.e. the number of responses in the chosen arm is at least `min.resp` and either none of the remaining arms exceed the `min.resp` threshold or the chosen (best) arm is better than the second best by at least `min.diff` responses (rate).

**References**

Simon R, Wittes RE, Ellenberg SS. (1985). Randomized phase II clinical trials. *Cancer Treat Rep* 69, 1375-1381.

**Examples**

```
pselect(18, c(0.2, 0.2, 0.2)) # selection when no difference i.e. type I error
pselect(18, c(0.2, 0.2, 0.4)) # selection probability
pselect(26, c(0.2, 0.2, 0.4), min.diff=2, min.resp=3)
pselect(c(27,54), c(0.5, 0.65), min.diff=0.05) # unequal sample size case
pselect(c(27,54), c(0.5, 0.65), min.diff=0.05, min.resp=c(14,27)) # unequal sample size ca
```

---

toxbdry

*Stopping rule for toxicity monitoring*

---

**Description**

Computes a stopping rule and its operating characteristics for toxicity monitoring based repeated significance testing.

**Usage**

```
toxbdry(pLo, pHi, n, cP0=0.1, cP1=0.9, ngrid=6, niter=5,
        priority=c("null", "alt"))
bdrycross.prob(n, r, ptox)
## S3 method for class 'toxbdry':
print(x, ...)
```

**Arguments**

pLo	the toxicity rate that is acceptable.
pHi	the toxicity rate that is too high and hence unacceptable.
n	vector of times (sample size) when toxicity is monitored.
r	vector of maximum acceptable toxicities corresponding to n.
ptox	the toxicity rates for which the operating characteristics are calculated.
cP0	boundary crossing probability under pLo i.e. type I error or the probability of declaring a treatment with toxicity rate pLo unacceptable.
cP1	boundary crossing probability under pHi i.e. power or the probability of declaring a treatment with toxicity rate pHi unacceptable.
ngrid	the number of toxicity rates from pLo to pHi for which the operating characteristics are computed.
niter	the number of iterations run to obtain the boundary.
priority	the error threshold to prioritize when the max sample size is too small to have both error thresholds satisfied. Default is the null i.e. error under pLo.
x	object returned by the function toxbdry.
...	additional arguments to print.

**Value**

the function returns a list with:

looks	when toxicity is monitored - same as input n.
lo.bdry	lower boundary is a vector of maximum acceptable number of toxicities corresponding the number of subjects in n. The boundary crossing probability for this is slightly above cP0.
hi.bdry	upper boundary is a vector of maximum acceptable number of toxicities corresponding the number of subjects in n. The boundary crossing probability for this is slightly below cP0.
bdry.oc	the operating characteristics i.e the toxicity rate, the probability of crossing, stopping (i.e. cross before the last observation) and the expected sample size for both the low (lo) and high (hi) boundaries.
bdry.alpha	the alpha levels for testing at each look for the two boundaries.

stopping for toxicity is done when the number of toxicities exceeds the boundary i.e. the boundary gives the maximum acceptable number.

**Examples**

```
toxbdry(0.2, 0.35, c(20,40,60,75))
toxbdry(0.2, 0.3, c(20,40,60,75), cP0=0.15, cP1=0.8)
toxbdry(0.1, 0.3, 2:30) # continuous monitoring
toxbdry(0.1, 0.3, 2:25, priority="alt") # prioritize cP1 error threshold
```

# Index

## \*Topic **design**

- fedesign, 4
- gsdesign, 5
- oc.twostage.bdry, 6
- ph2simon, 7
- ph2single, 8
- power.ladesign, 9
- pselect, 10
- toxbdry, 11

## \*Topic **htest**

- calogrank, 1
- permtests, 7

## \*Topic **survival**

- coxphCPE, 2
- coxphQuantile, 3

bdrycross.prob(*toxbdry*), 11

calogrank, 1  
coxphCPE, 2  
coxphQuantile, 3  
CPS.ssize(*fedesign*), 4

fe.mdor(*fedesign*), 4  
fe.power(*fedesign*), 4  
fe.ssize(*fedesign*), 4  
fedesign, 4

gsdesign, 5

jonckheere.test(*permtests*), 7

oc.twostage.bdry, 6

permlogrank(*permtests*), 7  
permtests, 7  
ph2simon, 7  
ph2single, 8  
plot.ph2simon(*ph2simon*), 7  
power.ladesign, 9  
print.ladesign(*power.ladesign*), 9

print.ph2simon(*ph2simon*), 7  
print.toxbdry(*toxbdry*), 11  
pselect, 10  
toxbdry, 11