Package ‘bpcp’

June 14, 2016

Type Package
Title Beta Product Confidence Procedure for Right Censored Data
Version 1.3.4
Date 2016-06-14
Author Michael P. Fay
Maintainer Michael P. Fay <mfay@niaid.nih.gov>
Depends stats
Suggests survival
Description Calculates nonparametric pointwise confidence intervals for the survival distribution for right censored data. Has two-sample tests for dissimilarity (e.g., difference, ratio or odds ratio) in survival at a fixed time. Especially important for small sample sizes or heavily censored data. Includes mid-p options.
License GPL (>= 2)
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2016-06-14 15:22:51

R topics documented:

   bpcp-package ............................................. 2
   betaMeldTest ......................................... 3
   bpcp2samp ............................................ 5
   bpcp2sampControl ..................................... 7
   bpcpControl .......................................... 8
   fixtdiff ............................................... 8
   kmci.object ......................................... 10
   kmciLR.object ....................................... 11
   kmtestALL ........................................... 12
   leuk .................................................. 16
   leuk2 ............................................... 17
**bpcp-package**

Pointwise Confidence Intervals Associated with the Survival Distribution for Right Censored Data

**Description**

The package has functions to give several different methods for calculating pointwise confidence intervals for a single survival distribution for right censored data. There is also a two-sample test for dissimilarity (measured by difference, ratio, or odds ratio) between two survival distributions at a fixed point in time.

The recommended confidence interval for a single sample is the beta product confidence procedure (using `bpcp`), and the recommended test for the two-sample test is the melded BPCP test (using `bpcp2samp`).

Other confidence intervals and two sample tests are included in the package primarily to compare them to the recommended ones. (And justify the recommendations).

Also included is a difference in medians test that applies only to non-censored data and is designed to guarantee coverage for all sample sizes (see `mdiffmedian.test`). The test makes no assumptions about the distributions, so that, unlike the Hodges-Lehmann method, tied data are allowed and a shift assumption is not needed.

**Details**

```r
Package: bpcp
Type: Package
Version: 1.3.4
Date: 2016-06-14
License: GPL2
LazyLoad: yes
```

The most important function for the single sample case is the `bpcp` which gives confidence intervals for the survival distribution for right censored data with nice small sample properties. That function creates an `kmcilR` object which has 6 methods: `summary` (create a data frame with estimates and confidence intervals), `plot` (plot Kaplan-Meier with confidence intervals), `lines` (add confidence intervals to a plot), `StCI` (pick out survival and confidence interval at specific times), `median` (pick out median and confidence interval on median), and `quantile` (pick out any quantile and its confidence interval). A mid-p option for `bpcp` is now available. It gives closer to nominal coverage than
the standard (midp=FALSE) BPCP. For details see Fay et al (2013) on the standard BPCP and Fay and Brittain (2016) on the mid-p option.

For the two-sample test see `bpcp2samp`. This test for equality reduces to Fisher’s exact test when there is no censoring. When there is censoring, the test is expected to maintain at least nominal coverage. For details see Fay et al (2015).

**Author(s)**

Michael P. Fay

Maintainer: Michael P. Fay <mfay@niaid.nih.gov>

**References**


**See Also**

`bpcp`

**Examples**

```r
data(leuk)
  ## since there are ties at time=6
  ## and the data are truncated to the nearest integer, use Delta=1
  bfit<-bpcp(leuk$time, leuk$status, Delta=1)
  ## plot Kaplan-Meier and 95 pct Beta Product Confidence Intervals
  plot(bfit, xlab="time (weeks)")
  ## details
  summary(bfit)
  quantile(bfit)
  StCI(bfit, 2)
```

---

**betaMeldTest**  
*Melded Beta Test*

**Description**

This function gives a two sample melded beta test together with the associated melded confidence intervals. It can be used when the confidence distributions (or upper and lower confidence distributions) for the one sample parameters are beta, and one is interested in either the difference, odds ratio, or ratio of those two one sample parameters. The betaMeldTest is usually called by `bpcp2samp`, and not called directly by the user.
Usage

\[
\text{betaMeldTest}(\text{betaParms1}, \text{betaParms2},
\text{nullparm} = \text{NULL},
\text{parmtype} = \text{c}("\text{difference}", "\text{oddsratio}", "\text{ratio}")
\text{conf.level} = 0.95, \text{conf.int} = \text{TRUE},
\text{alternative} = \text{c}("\text{two.sided}", "\text{less}", "\text{greater}")
\text{eps} = 10^{-8}, \text{dname} = "\text{"}, \text{estimate1} = \text{NA}, \text{estimate2} = \text{NA})
\]

Arguments

- **betaParms1**: a list of the beta parameters for group 1
- **betaParms2**: a list of the beta parameters for group 2
- **nullparm**: null value of the parameter of interest, default of NULL gives 0 if parmtype='difference' and 1 otherwise
- **parmtype**: parameter type for comparing the survival function of the two groups, either 'difference' 'ratio' or 'oddsratio'
- **conf.level**: confidence level, e.g., 0.95
- **conf.int**: logical, calculate confidence interval?
- **alternative**: character, either 'two.sided','less', or 'greater'
- **eps**: small value to make integration tractable
- **dname**: name describing data
- **estimate1**: estimate of mean for beta parameter of group 1 (statistic of htest object)
- **estimate2**: estimate of mean for beta parameter for group 2 (parameter of htest object)

Details

If the upper and lower confidence distributions for both samples are described by beta distributions, then you can create a CD test using this function. For example, if you have sample 1 is binomial with x (with 0<x<n) out of n positive responses, then the 100(1-alpha) confidence interval is qbeta(alpha/2, x, n-x+1) and qbeta(1-alpha/2, x+1, n-x). So the lower confidence distribution is beta with parameters a=x and b=n-x+1, and the upper CD is beta with parameters a=x+1 and b=n-x.

Value

an object of class 'htest'

Examples

\[
\text{fisher.test(matrix(c(4,5,2,22),2,2),alternative="greater")}
\]

betaMeldTest(
  betaParms1=list(alower=2,blower=22+1,aupper=2+1,bupper=22),
  betaParms2=list(alower=4,blower=5+1,aupper=4+1,bupper=5),
  alternative="greater",parmtype="oddsratio",
  estimate1=2/24,estimate2=4/9)
**bpcp2samp**

**Melded BPCP test**

**Description**
Tests for dissimilarity between two groups in their survival distributions at a fixed point in time. Can operationalize that dissimilarity as 'difference', 'ratio' or 'oddsratio'.

**Usage**

```r
bpcp2samp(timeL statusL groupL testtimeL
parmtype ] c(BdifferenceBL BoddsratioBL BratioBIL
nullparm ] NULL,
alternative ] c("two.sided", "less", "greater"),
conf.level ] 0.95,
midp=FALSE,
control = bpcp2sampControl())
```

**Arguments**

- `time`: time to event for each observation
- `status`: status of event time, 1 is observed, 0 is right censored
- `group`: group for test, should have two levels, to change order use as factor and change order of levels
- `testtime`: fixed time when you want to test for a difference
- `parmtype`: parameter type for comparing the survival function of the two groups, either 'difference', 'ratio' or 'oddsratio'
- `nullparm`: null value of the parameter of interest, default of NULL gives 0 if parmtype='difference' and 1 otherwise
- `alternative`: character, either 'two.sided', 'less', or 'greater'
- `conf.level`: confidence level, e.g., 0.95
- `midp`: logical, do mid-p tests and confidence intervals?
- `control`: list of control parameters, see `bpcp2sampControl`

**Details**
The melded confidence interval method is a very general procedure to create confidence intervals for the two sample tests by combining one sample confidence intervals. If $S_1$ and $S_2$ are the survival value at testtime from sample 1 (first value of group) and sample 2 (second value of group) respectively, then we can get confidence intervals on the $S_2 - S_1$ (parmtype='difference'), $S_2/S_1$ (parmtype='ratio'), or $(S_2/(1-S_2))/(S_1*(1-S_1))$ (parmtype='oddsratio').

The resulting melded CIs appear to guarantee coverage as long as the one sample confidence intervals from which the melded CIs are derived have guaranteed coverage themselves. So since we use the BPCP for the one sample intervals and they appear to guarantee coverage (see Fay, Brittain,
and Proschan, 2013), we expect the melded BPCP intervals to have at least nominal coverage. Note that when there is no censoring the melded CIs derived from the one-sample BPCPs, give matching inferences to Fisher’s exact test (i.e., give theoretically identical p-values) when testing the null hypothesis of equality (S1=S2). For details see Fay, Proschan and Brittain (2015).

The original melded CIs focused on combining one sample CIs that that guarantee coverage. We can apply the melding to other CIs as well, such as the mid-p style CIs. The mid-p CIs are not designed to guarantee coverage, but are designed to have close to the nominal coverage ‘on average’ over all the possible values of the parameters. The usual p-value is derived from Pr[ see observed data or more extreme under null], while the mid p-value version comes from (1/2) Pr[see obs data] + Pr[ see more extreme data]. Mid-p CIs come from inverting the test that uses the mid p-value instead of the usual p-value.

Value

A list with class “htest” containing the following components:

- statistic: estimate of S1, survival at testtime for group 1
- parameter: estimate of S2, survival at testtime for group 2
- p.value: p-value for the test
- conf.int: a confidence interval for the parameter determined by parmtype
- estimate: estimate of parameter determined by parmtype
- null.value: the specified null hypothesized value of the parameter determined by parmtype
- alternative: type of alternative with respect to the null.value, either 'two.sided', 'greater' or 'less'
- method: a character string describing the test
- data.name: a character string describing the parameter determined by parmtype

Author(s)

Michael P. Fay

References


Examples

data(leukR)
bpcp2samp(leukR$time, leukR$status, leukR$treatment, 35, parmtype=“ratio”)

bpcp2samp(leukR$time, leukR$status, leukR$treatment, 35, parmtype=“difference”)

**bpcp2sampControl**

Control function for *bpcp2samp*

---

**Description**

Call function to change any one of options, and outputs a list with all defaults except argument that you changed.

**Usage**

```r
bpcp2sampControl(Delta = 0, stype = "km", eps = 10^-8, 
nmc=10^6, method="mm.mc", seed=391291)
```

**Arguments**

- **Delta**
  - width of grouped confidence intervals, defaults to 0

- **stype**
  - type of survival estimate, either "km" for Kaplan-Meier or "mue" for median unbiased estimator

- **eps**
  - small value to make integration tractable

- **nmc**
  - number of Monte Carlo replications

- **method**
  - either 'mm.mc' (method of moments for one sample, meld with Monte Carlo) or 'mc.mc' (Monte Carlo for one sample and melding)

- **seed**
  - random number seed, if NULL do not set random number seed

**Details**

We set the seed by default, so that the same data set will always give the same results. If you are doing simulations, this setting of the seed will give problems. So use seed=NULL.

**Value**

A list containing the 6 arguments.

**See Also**

*bpcp2samp*

**Examples**

```r
bpcp2sampControl(Delta=1)
```
bpcpControl  

Inputs for adjusting numerical calculations in bpcp

Description

Function that returns a list of arguments.

Usage

bpcpControl(midpMMtol = .Machine$double.eps^0.25,  
seed=49911,  
tolerance=.Machine$double.eps^0.5)

Arguments

midpMMtol  value used for tol argument in uniroot call for calculating the midp method of moments method.
seed  seed for set.seed() when using Monte Carlo method. If is.null(seed) then do not set the seed.
tolerance  lowest positive value, such that abs(x-y)<tolerance treats x as equal to y. Used in bpcp for seeing if difference between times are equal to Delta or not.

Details

When doing simulations on the Monte Carlo method, set seed=NULL. Then the seed will not be set at each replication. The default is to set the seed to 49911, so two analyses of the same data on the same version of R will give identical results.

Value

A list with components named as the arguments.

fixtdiff  

Two sample test for Difference in Survival at Fixed Time

Description


Usage

fixtdiff(time,status,group, testtime,  
trans=c("identity","cloglog","log"),  
varpooled=TRUE, correct=FALSE, doall=FALSE)
Arguments

- **time**: time to event for each observation
- **status**: status of event time, 1 is observed, 0 is right censored
- **group**: group for test, should have two levels, to change order use as factor and change order of levels
- **testtime**: fixed time when you want to test for a difference
- **trans**: type of transformation, one of 'identity', 'cloglog' or 'log'
- **varpooled**: logical, pool the variance?
- **correct**: logical, do continuity correction? Continuity correction for when trans='identity' and varpooled (see Warning)
- **doall**: logical, do all transformations and corrections

Details

This function provides p-values for the two sample tests that the survival distributions are equal at time *testtime*. The tests are asymptotically normal tests and are described in Klein, et al (2007). These functions are mostly for simulations to evaluate the melded BPCP tests, see `bpcp2samp` and Fay et al (2015).

Value

A list with the following components:

- **plo**: one-sided p-value, alternative: \( S_1(\text{testtime}) > S_2(\text{testtime}) \)
- **phi**: one-sided p-value, alternative: \( S_1(\text{testtime}) < S_2(\text{testtime}) \)
- **p2**: two-sided p-value, \( \min(1,2*plo,2*phi) \)

Warning

Continuity correction derived from the case with no censoring (see Fleiss et al 3rd edition, pp. 50-55). May not make sense when there is censoring. Use at own risk.

Author(s)

Michael P. Fay

References

Examples

```r
data(leuk2)
# Note that since the Kaplan-Meier survival at time=35 goes to
# zero for one group, the results for the log and cloglog
# transformations are undefined
fixtdiff(leuk2$time, leuk2$status, leuk2$treatment, 35, doall=TRUE)
```

---

**kmci.object**

*Kaplan-Meier (Survival Curve) Confidence Interval Object*

### Description

The `kmci` class is returned by the functions `kmciTG` or `kmciSW`. The class represents a fitted survival curve with pointwise confidence intervals.

Unlike the `kmciLR` class, which allows for confidence intervals to change at any time point, the `kmci` class only has the confidence intervals change at observed failures.

Objects of this class has methods for the functions `summary`, `plot`, `lines`.

### Arguments

- **time**: the time points of observed failures (assumed `surv` and lower and upper steps that these times)
- **cens**: time points where there is censoring but no observed failure
- **surv**: the estimate of survival at time $t+0$. This is a vector.
- **upper**: upper confidence limit for the survival curve.
- **lower**: lower confidence limit for the survival curve.
- **conf.level**: the level of the confidence limits, e.g., 0.95.

### Structure

The following components must be included in a legitimate `kmci` object.

### See Also

- `kmciLR.object`  
- `plot.kmci`  
- `summary.kmci`  
- `StCI.kmci`  
- `median.kmci`  
- `quantile.kmci`
The kmciLR class returned by the functions `bpcp` or `kmciBorkowf`, and represents a fitted survival curve with pointwise confidence intervals.

The kmciLR class allows for confidence intervals to change at any time point, while the kmci class only has the confidence intervals change at observed failures.

Objects of this class has methods for the functions `summary`, `plot`, `lines`.

**Arguments**

- `cens` time points where there is censoring but no observed failure
- `surv` the estimate of survival in the interval described by L and R. This is a vector.
- `upper` upper confidence limit for the survival curve in the interval described by L and R.
- `lower` lower confidence limit for the survival curve in the interval described by L and R.
- `L` vector of left ends of interval associated with lower and upper
- `Lin` vector of logicals, should left end of interval be included?
- `R` vector of right ends of interval associated with lower and upper
- `Rin` vector of logicals, should right end of interval be included?
- `Interval` character vector describing intervals
- `stype` character vector giving type of survival estimate, either 'km' or 'mue'
- `conf.level` the level of the confidence limits, e.g., 0.95.

**Structure**

The following components must be included in a legitimate kmciLR object.

**See Also**

`plot.kmci`, `summary.kmci`, `StCI.kmci`, `median.kmci`, `quantile.kmci`
**kmtestALL**

*Pointwise confidence intervals for survival for right censored data.*

**Description**

These functions give several different methods for calculating pointwise confidence intervals for the survival distribution for right censored data. The recommended confidence intervals are the beta product ones given by `bpcp`.

The other confidence intervals are included primarily to show that the beta product confidence procedure (using `bpcp`) has better coverage than the best alternatives. See details for a description of all the methods.

**Usage**

```r
bpcp(time, status, nmc=0, alpha=.05, Delta=0, stype="km", midp=FALSE, monotonic=NULL, control=bpcpControl())
```

```r
kmciBorkowf(time, status, type="log", alpha = 0.05)
kmtestBoot(time, status, tstar, pstar, M = 1000, alpha = 0.05)
kmtestConstrainBoot(time, status, tstar, pstar, M = 1000, alpha = 0.05)
kmtestConstrainBeta(time, status, tstar, pstar, alpha=.05)
kmciSW(time, status, alpha = 0.05)
kmciTG(time, status, alpha = 0.05)
kmciITG(time, status, tstar, alpha = 0.05)
```

```r
kmtestALL(time, status, t0, S0, cens=NULL, M=1000, NMC=10^5, alpha=0.05)
```

**Arguments**

- `time`: time to event or censoring
- `status`: status vector, 1 is event, 0 is censoring
- `alpha`: 1- conf.level
- `nmc`: number of Monte Carlo replications from each beta distribution, nmc=0 means use method of moments for beta parameters instead
- `NMC`: same as nmc
- `Delta`: width of grouped confidence intervals, defaults to 0 (rarely need to change this, even with ties, see details)
- `stype`: type of survival estimate, either "km" for Kaplan-Meier or "mue" for median unbiased estimator
- `midp`: logical, calculate the mid-p type of interval?
- `monotonic`: logical, force lower and upper confidence limits to be monotonic over time? If NULL: nmc=0 gives TRUE, nmc>0 gives FALSE
control: list with arguments for adjusting numeric calculation. Generally does not need to be changed. See `bpcpControl`

tstar: time to test survival distribution

pstar: null survival distribution

M: number of bootstrap replications

t0: null hypothesis time for survival test

S0: null hypothesis value of survival at t0

cens: vector of censoring times (even those with failures before it), used for Binomial test. If NULL gives NA for binom test

type: see details

Details

The standard beta product confidence procedure (i.e., with midp=FALSE) will give pointwise confidence intervals for right censored data with the following properties. When there is no censoring or Progressive Type II censoring the BPCP guarantees central coverage (e.g., the error rate on either side of the 95 percent confidence interval is guaranteed to be less than 2.5 percent). For general independent censoring the BPCP is asymptotically equivalent to standard methods such as the normal approximation with Greenwood variance, and hence the BPCP (as with the other confidence interval given here) goes to the correct confidence interval for any t.

There is also a mid-p version of the BPCP. The BPCP is derived from using the known distribution of the failure times, and acting conservatively between the failure times (see Fay, Brittain, and Proschan, 2013 for details). Instead of acting conservatively between the failure times, the midp=TRUE version combines the distributions for the previous failure and the future failure time (see Fay and Brittain, 2016).

For description of how `bpcp` with different values of Delta works, see “Beta Product Confidence Intervals for Discrete Failure Times” vignette (especially Section 2.2). Note especially that confidence intervals exactly at the failure times when Delta=0 are handled differently before Version 1.3.0 than from Version>=1.3.0. For users not interested in details who only want to know the recommended confidence intervals on right censored data when ties are allowed, we recommend the `bpcp` function version 1.3.0 or greater using the default Delta=0 argument. That recommendation will give pointwise confidence intervals that treats ties similarly to the way that the Kaplan-Meier estimator treats ties, and hence will give confidence intervals that enclose the Kaplan-Meier estimate.

Now we describe the other methods. In general the functions are of three naming types: `kmtestXX`, `kmci1XX` and `kmciXX`, where XX changes for different methods. Functions `kmtestXX` only test whether S(tstar)=pstar and return a vector of 1s for reject and 0s for fail to rejecting either of the one-sided or the two-sided hypotheses. Functions `kmci1XX` only give confidence intervals at S(tstar), while `kmciXX` give confidence intervals for all values of t. The standard methods calculate the confidence intervals at the observed failure times and carry them forward (e.g., kmciTG, kmciSW) and the results are objects of class kmci. More involved methods allow confidence intervals to change after censored objects (e.g., kmciBorkowf, bpcp) and the results are objects of class kmciLR.

The function `kmtestBoot` tests S(tstar)=pstar using the nonparametric bootstrap (sampling vectors of (time,status) with replacement) with the percentile method as described in Efron (1981). The function `kmtestConstrainBoot` and `kmtestConstrainBeta` tests S(tstar)=pstar using the constrained Bootstrap or constrained Beta method described in Barber and Jennison (1999).
The function `kmci1TG` does a confidence interval only at \( t_{\text{star}} \), while `kmciTG` does a confidence interval at all the observed event times. The method can be derived as a likelihood ratio test and is described in Thomas and Grunkemeier (1975). It has asymptotically correct coverage, which is rigorously proved in Murphy (1995). You can also think of the method as the empirical likelihood applied to the survival distribution for right censored data (see Owen, 2001, p. 144-145).

The function `kmcisw` calculates confidence intervals using Edgeworth expansions as described in Strawderman and Wells (1997). Note, Strawderman, Parzen and Wells (1997) is easier to understand than Strawderman and Wells (1997).

Borkowf (2005) creates confidence intervals for the Kaplan-Meier survival estimate for right censored data. He allows the confidence interval to change at censoring times as well as at failure times.

Four types of confidence intervals may be selected. The asymptotic normal approximation (type="norm"), the shifted K-M estimate with normal approximation (type="norms"), the log transformed normal approximation using the delta method (type="log"), and the log transformed normal approximation using the delta method with the shifted K-M (type="logs").

The function `kmtestALL` performs hypothesis tests on all the methods except the unconstrained bootstrap method (unless M=0 then it does not test the constrained bootstrap method either). The output is a matrix with three columns with a value of 1 representing either (1) rejection for two-sided test implying the estimate is greater than the null, (2) rejection for two-sided test implying the estimate less than the null, or (3) any rejection of the two-sided test. Each row represents a different test.

The `kmci` or `kmciLR` classes have the following methods: "plot","lines","summary","quantile", and "median". Additionally, you can pull out survival and confidence intervals from these objects at specific times using "StCI".

**Value**

The functions return an object of class either `kmci` or `kmciLR` see details). both `kmci` and `kmciLR` objects are lists, and both contain elements

- `surv`: survival distribution in interval
- `lower`: lower pointwise confidence limit in interval
- `upper`: upper pointwise confidence limit in interval

Additionally the `kmci` objects have an element

- `time`: time of survival or confidence interval

while the `kmciLR` have intervals represented by the four elements

- `L`: left endpoint of interval
- `Lin`: logical vector, include left endpoint?
- `R`: right endpoint of interval
- `Rin`: logical vector, include right endpoint?

And results from `bpcp` additionally have an element

- `betaParms`: list with 4 elements of beta parameters associated with the CIs: alower, blower, aupper, bupper
Author(s)

Michael Fay

References


Thomas and Grunkemeier (1975) JASA 70: 865-871.

See Also

The \texttt{kmci} and \texttt{kmciLR} objects have methods: \texttt{plot}, \texttt{lines}, \texttt{summary}, \texttt{quantile}, and \texttt{median}, \texttt{StCI}.

Examples

```r
library(bpcp)
data(leuk)

### Recommended method is bpcp
### since the data are truncated to the nearest integer
### use Delta=1 option
out<-bpcp(leuk$time,leuk$status,Delta=1)
summary(out)
median(out)
plot(out)

### Borkowf 2005 method
norm<-kmciBorkowf(leuk$time,leuk$status,type="norm")
norms<-kmciBorkowf(leuk$time,leuk$status,type="norms")
### check Table VII of Borkowf
I<-c(1,2,3,5,7,8,9,11,13,15,17,19,21,23,25,27,29,31,33)
round(data.frame(lowerNorm=norm$lower[I],
               upperNorm=norm$upper[I],lowerNorms=norms$lower[I],
               upperNorms=norms$upper[I],row.names=norm$Interval[I]),3)
```
### Strawderman and Wells (1997) method
```
swci<-kmciSW(leuk$time,leuk$status)
summary(swci)
```

### Thomas and Grunkemeier 1975 method
```
x<-kmciTG(leuk$time,leuk$status)
summary(x)
```

## Compare to Table 1, Sample 2, of Thomas and Grunkemeier (1975)
```
StCI(x,c(10,20))
```

---

### Description

This is only the 21 patients who received 6-mercaptopurine (6-MP). There were 21 patients who got placebo (see `leuk` for complete data).

See also Borkowf (2005)

### Usage

```
data(leuk)
```

### Format

A data frame with 21 observations on the following 2 variables.

- **time**: time in remission (in weeks)
- **status**: event status, 1 is relapse, 0 is censored

### References


### See Also

`leuk` for complete data.

### Examples

```
data(leuk)
```

Description
In this study there were 21 pairs of subjects, and within each pair one subject received 6-mercaptopurine (6-MP) and one got placebo. The data are right censored.

See also Gehan (1965) who used the data ignoring the pairing so that he could illustrate his famous two-sample (non-paired) rank test.

Usage
data(leuk2)

Format
A data frame with 42 observations on the following variables.

- time  time in remission (in weeks)
- status  event status, 1 is relapse, 0 is censored
- treatment  treatment group: either 'placebo' or '6-MP'
- pair  pair id number

References

See Also
leuk is only the treated group

Examples
data(leuk2)
mdiffmedian.test  Melded Difference in Medians Test

Description

Tests for a difference in two medians. No assumptions about the two distributions are needed (may be discrete with ties allowed, no shift assumption is required). Uses the melded confidence interval derived from the one sample confidence intervals associated with the sign test (a version that allows for ties). Derivation of the test does not require large samples, and confidence intervals are intended to guarantee coverage regardless of sample size.

Usage

mdiffmedian.test(x1, x2, nulldiff = 0,
    alternative = c("two.sided", "less", "greater"),
    conf.level = 0.95)

Arguments

x1  vector of numeric responses from group 1
x2  vector of numeric responses from group 2
nulldiff difference in medians under the null, median(x2)-median(x1)
alternative a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level confidence level of the interval.

Details

The melded confidence interval method is a general method for combining two one-sample confidence intervals (CIs). In this function, we use the melded CI method on the two one-sample CIs from the sign test that allows for ties. This creates CIs for the difference in medians that requires very few assumptions. In particular, ties are allowed and no shift assumption is needed. For details see Fay, Proschan and Brittain (2014).

Value

a list of class 'htest' with elements:

statistic median of x1
parameter median of x2
p.value p-value of the test
conf.int confidence interval for the difference in medians
estimate median(x2)-median(x1)
null.value null hypothesis value for difference in medians
alternative type of alternative hypothesis
Note

This function does not allow censoring. Also, there is a price for not needing large samples nor assumptions about the distributions: if you do not have enough data, your confidence intervals may be the entire real line. For example, if you have continuous data with equal sample sizes in both groups, then if you have 6 or fewer observations in each group, then the 95 percent confidence interval on the difference in medians will be (-Inf,Inf).

Author(s)

Michael P. Fay

References


Examples

```r
set.seed(1)
trtA<-rpois(20,1.5)
trtB<-rpois(23,5.5)
mdiffmedian.test(trtA,trtB)
```

---

**plot.kmcilr**  
*Plot and lines methods for kmci and kmcilr objects.*

**Description**

Plots survival curves and/or confidence intervals.

**Usage**

```r
## S3 method for class 'kmci'
plot(x, ...)
```

```r
## S3 method for class 'kmcilr'
plot(x, XLAB = "time", YLAB = "Survival", YLIM = c(0, 1),
     ciLTY = 2, ciCOL = gray(0.8), mark.time = NULL, linetype = "both", ...)
```

```r
## S3 method for class 'kmcilr'
lines(x, lty = c(2, 1), col = c(gray(0.8), gray(0)),
      linetype = "both", mark.time = NULL, ...)
```
## S3 method for class 'kmci'
lines(x, ...)

### Arguments
- **x**: kmci or kmciLR object (created by functions described in `kmtestALL`)
- **XLAB**: label for x axis
- **YLAB**: label for y axis
- **YLIM**: limits for y axis
- **ciLTY**: lty (line type) for confidence intervals
- **ciCOL**: col (color) for confidence intervals
- **col**: vector of colors, first element used for ci second for survival curve
- **lty**: vector of line types, first element used for ci second for survival curve
- **mark.time**: put hash marks for censored objects (default puts marks of stype="km" but not if stype="mue")
- **linetype**: character, which lines to draw: either 'both', 'surv' or 'ci'
- **...**: Extra parameters to be passed. Any argument in `plot.kmciLR` can be passed from `plot.kmci`, similarly for line. Other parameters are usually graphical parameters passed to plot and segment calls within function.

### Examples
```r
data(leuk)
## kmciTG creates kmci object
fitTG<-kmciTG(leuk$time,leuk$status)
plot(fitTG)
## bpcp creates kmciLR object
fitBP<-bpcp(leuk$time,leuk$status)
lines(fitBP,lwd=3,lty=1,col=gray(.5))
legend(0,.2,legend=c("Kaplan-Meier","Thomas-Grunkemeier 95 pct CI","Beta Product 95 pct CI"),
       lwd=c(1,1,3),lty=c(1,2,1),col=c(gray(0),gray(.8),gray(.5)))
```

---

**quantile.kmciLR**  
**Quantiles or Medians from kmci or kmciLR objects.**

### Description
Get quantiles or median with the associated confidence intervals from a kmci or kmciLR object.
sclerosis

Usage

```r
## S3 method for class 'kmciLR'
quantile(x, probs = c(0.25, 0.5, 0.75), ...)
## S3 method for class 'kmci'
quantile(x, probs = c(0.25, 0.5, 0.75), ...)
## S3 method for class 'kmciLR'
median(x, ...)
## S3 method for class 'kmci'
median(x, ...)
```

Arguments

- `x` : a kmci or kmciLR object
- `probs` : vector of probability to calculate quantiles
- `...` : parameters passed

Value

matrix same number of rows as probs and with 4 columns
- `S(q)` : probs, survival estimate at quantile
- `q` : quantile
- `lower` : lower confidence limit of q
- `upper` : upper confidence limit of q

Examples

```r
data(leuk)
## kmciTG creates kmci object
fitTG<-kmciTG(leuk$time,leuk$status)
quantile(fitTG)
## bpcp creates kmciLR object
fitBP<-bpcp(leuk$time,leuk$status)
median(fitBP)
```

---

sclerosis


Description

Severe systemic sclerosis is a serious autoimmune disease affecting multiple organs including the heart, lungs, kidney, and skin. Between 1997 and 2005, a cohort of 34 patients was enrolled in a single arm pilot study of high-dose immunosuppressive therapy and autologous hematopoietic cell transplantation.
Usage

data(sclerosis)

Format

A data frame with 34 observations on the following 3 variables.

day  time to death or censoring, in days
year  time to death or censoring, in years (day/365.25)
status 0 is censored, 1 is event

References


Examples

data(sclerosis)
plot(bpcp(sclerosis$year,sclerosis$status))

StCI

Get survival and confidence interval at t from kmci, kmciLR, or survfit object

Description

Just picks out the survival function and confidence interval in a different way depending on the type of object.

Usage

## Default S3 method:
StCI(x,tstar, afterMax = "continue", ...)

## S3 method for class 'kmciLR'
StCI(x,tstar, ...)


Arguments

- **x**: a kmci or kmciLR object
- **tstar**: a vector of times that you want survival and CI values
- **afterMax**: character, what to do after tmax (see details)
- **...**: further arguments to be passed to or from methods.

Details

Since the Kaplan-Meier estimator is undefined after the last observation if it is censored and many confidence interval methods are not defined there either, we need to explicitly define what to do. (For objects of the kmciLR class, the confidence intervals are defined over the positive real line and the afterMax is ignored.) The afterMax has four options for this: 'continue' (keep surv and ci values the same as the last calculated one), 'zero' (surv and lower go to zero, upper stays same), 'zeroNoNA' (surv and lower go to zero, upper stays same unless it is NA, then it takes on the last non-missing upper value), 'half' (surv goes to half value, lower goes to zero, upper stays same).

Value

The function StCI returns a data frame with the following variables. (It also has an attribute: 'conf.level').

- **time**: this is tstar
- **survival**: survival at tstar
- **lower**: lower confidence limit at tstar
- **upper**: upper confidence limit at tstar

Author(s)

Michael Fay

See Also

- kmci, kmciLR

Examples

```r
data(leuk)
## compare to table 1 of Thomas and Grunkmeier (1975)
StCI(kmciT0(leuk$time, leuk$status), c(10, 20))
```
summary.kmciLR

Summary method for kmci or kmciLR object.

Description

Creates a data frame with time (for kmci) or time interval (for kmciLR), survival, lower and upper pointwise confidence intervals.

Usage

## S3 method for class 'kmciLR'
summary(object, ...)

## S3 method for class 'kmci'
summary(object, ...)

Arguments

- object: kmci or kmciLR object
- ...: extra arguments

Value

creates a data frame. See description.

Examples

data(leuk)
## kmciTG creates kmci object
fitTG<-kmciTG(leuk$time,leuk$status)
summary(fitTG)
## bpcp creates kmciLR object
fitBP<-bpcp(leuk$time,leuk$status)
summary(fitBP)
Index

+Topic \textasciitilde kwd1
  bpcpRsamp, 5
+Topic \textasciitilde kwd2
  bpcpRsamp, 5
+Topic datasets
  leuk, 16
  leukR, 17
  sclerosis, 21
+Topic htest
  betaMeldTest, 3
  fixtdiff, 8
  kmtestALL, 12
  mdiffmedian.test, 18
  StCI, 22
+Topic misc
  bpcp2sampControl, 7
+Topic nonparametric
  bpcp-package, 2
  kmtestALL, 12
  StCI, 22
+Topic optimize
  bpcpControl, 8
+Topic package
  bpcp-package, 2
+Topic survival
  bpcp-package, 2
  kmci.object, 10
  kmciLR.object, 11
  kmtestALL, 12
  plot.kmciLR, 19
  quantile.kmciLR, 20
  StCI, 22
  summary.kmciLR, 24

betaMeldTest, 3
bpcp, 2, 3, 11
bpcp (kmtestALL), 12
bpcp-package, 2
bpcp2samp, 2, 3, 5, 7, 9
bpcp2sampControl, 5, 7

bpcpControl, 8, 13
fixtdiff, 8

kmci, 11, 14, 15, 23
kmci (kmci.object), 10
kmci.object, 10
kmciTG (kmtestALL), 12
kmciBorkowf, 11
kmciBorkowf (kmtestALL), 12
kmciLR, 10, 14, 15, 23
kmciLR (kmciLR.object), 11
kmciLR.object, 10, 11
kmciSW, 10
kmciSW (kmtestALL), 12
kmciTG, 10
kmciTG (kmtestALL), 12
kmtestALL, 12, 20
kmtestBoot (kmtestALL), 12
kmtestConstrainBeta (kmtestALL), 12
kmtestConstrainBoot (kmtestALL), 12

leuk, 16, 17
leuk2, 16, 17
lines, 14, 15
lines.kmci (plot.kmciLR), 19
lines.kmciLR (plot.kmciLR), 19

mdiffmedian.test, 2, 18
median, 14, 15
median.kmci, 10, 11
median.kmci (quantile.kmciLR), 20
median.kmciLR (quantile.kmciLR), 20

plot, 14, 15
plot.kmci, 10, 11
plot.kmci (plot.kmciLR), 19
plot.kmciLR, 19

quantile, 14, 15
quantile.kmci, 10, 11
quantile.kmci (quantile.kmciLR), 20
quantile.kmciLR, 20

tsclerosis, 21
StCI, 14, 15, 22
StCI.kmci, 10, 11
summary, 14, 15
summary.kmci, 10, 11
summary.kmci (summary.kmciLR), 24
summary.kmciLR, 24