Package ‘survcomp’

January 8, 2016

Type Package
Title Performance Assessment and Comparison for Survival Analysis
Version 1.20.0
Date 2015-07-21
Description R package providing functions to assess and to compare the performance of risk prediction (survival) models.
biocViews GeneExpression, DifferentialExpression, Visualization
Author Benjamin Haibe-Kains, Markus Schroeder, Catharina Olsen, Christos Sotiriou, Gianluca Bon- tempi, John Quackenbush
Maintainer Benjamin Haibe-Kains <benjamin.haibe.kains@utoronto.ca>, Markus Schroeder <markus.schroeder@ucdconnect.ie>, Catharina Olsen <colsen@ulb.ac.be>
Depends survival, prodlim, R (>= 2.10)
Suggests Hmisc, CPE, clinfun, Biobase
Imports ipred, SuppDists, KernSmooth, survivalROC, bootstrap, grid, rmeta
License Artistic-2.0
URL http://www.pmgenomics.ca/bhklab/
NeedsCompilation yes

R topics documented:

survcomp-package .................................................. 2
breastCancerData .................................................... 4
censor.time .......................................................... 6
cindex.comp ......................................................... 7
cindex.comp.meta .................................................. 8
combine.est ......................................................... 10
combine.test ....................................................... 11
concordance.index .................................................. 12
cvpl ................................................................. 14
survcomp-package

Performance Assessment and Comparison for Survival Analysis

Description

Functions to perform the performance assessment and comparison of risk prediction (survival) models.

Details

Package: survcomp
Type: Package
Version: 1.20.0
Date: 2015-07-21
License: Artistic-2.0
Author(s)

Benjamin Haibe-Kains
- Bioinformatics and Computational Genomics Laboratory, Princess Margaret Cancer Center, University Health Network, Toronto, Ontario, Canada

http://www.pmgenomics.ca/bhklab/

Former labs:
- Computational Biology and Functional Genomics Laboratory, Dana-Farber Cancer Institute, Boston, MA, USA

http://compbio.dfci.harvard.edu/
- Center for Cancer Computational Biology, Dana-Farber Cancer Institute, Boston, MA, USA

http://cccb.dfci.harvard.edu/index.html
- Machine Learning Group (MLG), Universite Libre de Bruxelles, Bruxelles, Belgium

http://www.ulb.ac.be/di/mlg/
- Breast Cancer Translational Research Laboratory (BCTRL), Institut Jules Bordet, Bruxelles, Belgium


Markus Schroeder
- UCD School of Biomolecular and Biomedical Science, Conway Institute, University College Dublin, Belfield, Dublin, Ireland

http://bioinfo-casl.ucd.ie/cib/people/students/markusschroder

Maintainer:
Benjamin Haibe-Kains
<benjamin.haibe.kains@utoronto.ca>
Markus Schroeder
<markus.schroeder@ucdconnect.ie>

See Also

survival, Hmisc, Design, prodlim, survivalROC, ipred, bootstrap, CPE, clinfun
breastCancerData  

Sample data containing six datasets for gene expression, annotations and clinical data.

Description

This dataset contains a subset of the gene expression, annotations and clinical data from 6 different datasets (see section details). The subsets contain the seven gene signature introduced by Desmedt et al. 2008.

Usage

data(breastCancerData)

Format

Six ExpressionSets. Example for ‘mainz7g’: eSet with 7 features and 200 samples, containing:

- `exprs(mainz7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a technology (single-channel, oligonucleotides).
- `fData(mainz7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a.
- `pData(mainz7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(mainz7g)`: MIAME object containing information about the dataset.
- `annotation(mainz7g)`: Name of the affy chip.

Details

This dataset represents subsets of the studies published by Schmidt et al. 2008 [mainz7g], Wang et al. 2005 and Minn et al. 2007 [vdx7g], Miller et al. 2005 [upp7g], Sotiriou et al. 2006 [unt7g], Desmedt et al. 2007 and TRANSBIG [transbig7g], van’t Veer et al. 2002 and van de Vijver et al. 2002 [nki7g]. Each subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as uPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

Source


References


Examples

```r
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)

## Example for the mainz7g dataset

## show the first 5 columns of the expression data
eprs(mainz7g)[,1:5]
## show the first 6 rows of the phenotype data
head(pData(mainz7g))
## show first 20 feature names
featureNames(mainz7g)
## show the experiment data summary
experimentData(mainz7g)
## show the used platform
annotation(mainz7g)
## show the abstract for this dataset
abstract(mainz7g)
```

---

censor.time  

*Function to artificially censor survival data*

Description

The function censors the survival data at a specific point in time. This is is useful if you used datasets having different follow-up periods.

Usage

```r
censor.time(surv.time, surv.event, time.cens = 0)
```

Arguments

- `surv.time`: vector of times to event occurrence
- `surv.event`: vector of indicators for event occurrence
- `time.cens`: point in time at which the survival data must be censored

Value

- `surv.time.cens`: vector of censored times to event occurrence
- `surv.event.cens`: vector of censored indicators for event occurrence

Author(s)

Benjamin Haibe-Kains
**Examples**

```r
set.seed(12345)
stime <- rexp(30)
cens <- runif(30, 0.5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
censor.time(surv.time=stime, surv.event=sevent, time.cens=1)
```

---

**cindex.comp**

*Function to compare two concordance indices*

**Description**

This function compares two concordance indices computed from the same data by using the function `concordance.index`. The statistical test is a Student t test for dependent samples.

**Usage**

```r
cindex.comp(cindex1, cindex2)
```

**Arguments**

- **cindex1**: first concordance index as returned by the `concordance.index` function.
- **cindex2**: second concordance index as returned by the `concordance.index` function.

**Details**

The two concordance indices must be computed from the same samples (and corresponding survival data). The function uses a Student t test for dependent samples.

**Value**

- **p.value**: p-value from the Student t test for the comparison cindex1 > cindex2.
- **cindex1**: value of the first concordance index.
- **cindex2**: value of the second concordance index.

**Author(s)**

Benjamin Haibe-Kains

**References**

See Also

concordance.index.

Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
size <- rexp(100)
stime <- rexp(100)
cens <- runif(100, 5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
c1 <- concordance.index(x=age, surv.time=stime, surv.event=sevent,
                          method="noether")
c2 <- concordance.index(x=size, surv.time=stime, surv.event=sevent,
                          method="noether")
cindex.comp(c1, c2)
```

---

**cindex.comp.meta**  
*Function to compare two concordance indices*

**Description**

This function compares two lists of concordance indices computed from the same survival data by using the function `concordance.index`. The statistical test is a Student t test for dependent samples.

**Usage**

```r
cindex.comp.meta(list.cindex1, list.cindex2, hetero = FALSE)
```

**Arguments**

- `list.cindex1`: first list of concordance indices as returned by the `concordance.index` function.
- `list.cindex2`: second list of concordance indices as returned by the `concordance.index` function.
- `hetero`: if TRUE, a random effect model is use to compute the meta-estimators. Otherwise a fixed effect model is used.

**Details**

In meta-analysis, we estimate the statistic of interest in several independent datasets. It results a list of estimates such as list of concordance indices. The two lists of concordance indices must be computed from the same samples (and corresponding survival data). The function computes a meta-estimator for the correlations between the two scores and uses a Student t test for dependent samples.
Value

- p.value: p-value from the Student t test for the comparison \( cindex_1 > cindex_2 \).
- cindex1: meta-estimator of the first concordance index.
- cindex2: meta-estimator of the second concordance index.

Author(s)

Benjamin Haibe-Kains

References


See Also

- `concordance.index`

Examples

```r
#first dataset
set.seed(12345)
age <- rnorm(100, 50, 10)
size <- rexp(100,1)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
c1.1 <- concordance.index(x=age, surv.time=stime, surv.event=sevent,
                         method="noether")
c2.1 <- concordance.index(x=size, surv.time=stime, surv.event=sevent,
                         method="noether")
#second dataset
set.seed(54321)
age <- rnorm(110, 53, 10)
size <- rexp(110,1.1)
stime <- rexp(110)
cens <- runif(110,.55,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
c1.2 <- concordance.index(x=age, surv.time=stime, surv.event=sevent,
                         method="noether")
c2.2 <- concordance.index(x=size, surv.time=stime, surv.event=sevent,
                         method="noether")
cindex.comp.meta(list.cindex1=list("cindex.age1"=c1.1, "cindex.age2"=c1.2),
                 list.cindex2=list("cindex.size1"=c2.1, "cindex.size2"=c2.2))
```
combine.est

Function to combine estimates

Description

The function combines several estimators using meta-analytical formula to compute a meta-estimate.

Usage

```r
combine.est(x, x.se, hetero = FALSE, na.rm = FALSE)
```

Arguments

- `x`: vector of estimates
- `x.se`: vector of standard errors of the corresponding estimates
- `hetero`: TRUE is the heterogeneity should be taken into account (random effect model), FALSE otherwise (fixed effect model)
- `na.rm`: TRUE if the missing values should be removed from the data, FALSE otherwise

Value

- `estimate`: meta-estimate
- `se`: standard error of the meta-estimate

Author(s)

Benjamin Haibe-Kains

References


See Also

- test.hetero.est

Examples

```r
set.seed(12345)
x1 <- rnorm(100, 50, 10) + rnorm(100, 0, 2)
m1 <- mean(x1)
se1 <- sqrt(var(x1))
x2 <- rnorm(100, 75, 15) + rnorm(100, 0, 5)
m2 <- mean(x2)
se2 <- sqrt(var(x2))

# fixed effect model
```
**combine.test**

```r
combine.est(x=c(m1, m2), x.se=c(se1, se2), hetero=FALSE) # random effect model
combine.est(x=c(m1, m2), x.se=c(se1, se2), hetero=TRUE)
```

---

### Description

The function combines several p-value estimated from the same null hypothesis in different studies involving independent data.

### Usage

```r
combine.test(p, weight, method = c("fisher", "z.transform", "logit"),
             hetero = FALSE, na.rm = FALSE)
```

### Arguments

- **p**: vector of p-values
- **weight**: vector of weights (e.g. sample size of each study)
- **method**: `fisher` for the Fisher's combined probability test, `z.transform` for the Z-transformed test, `logit` for the weighted Z-method
- **hetero**: TRUE is the heterogeneity should be taken into account, FALSE otherwise
- **na.rm**: TRUE if the missing values should be removed from the data, FALSE otherwise

### Details

The p-values must be one-sided and computed from the same null hypothesis.

### Value

- **p-value**

### Author(s)

Benjamin Haibe-Kains

### References


### See Also

- `test.hetero.test`
Examples

```r
p <- c(0.01, 0.13, 0.07, 0.2)
w <- c(100, 50, 200, 30)

# with equal weights
combine.test(p=p, method="z.transform")

# with p-values weighted by the sample size of the studies
combine.test(p=p, weight=w, method="z.transform")
```

concordance.index

*Function to compute the concordance index for survival or binary class prediction*

Description

Function to compute the concordance index for a risk prediction, i.e. the probability that, for a pair of randomly chosen comparable samples, the sample with the higher risk prediction will experience an event before the other sample or belongs to a higher binary class.

Usage

```r
concordance.index(x, surv.time, surv.event, cl, weights, comppairs=10, strat, alpha = 0.05, outx = TRUE)
```

Arguments

- `x`: a vector of risk predictions.
- `surv.time`: a vector of event times.
- `surv.event`: a vector of event occurrence indicators.
- `cl`: a vector of binary class indicators.
- `weights`: weight of each sample.
- `comppairs`: threshold for comparable patients.
- `strat`: stratification indicator.
- `alpha`: alpha level to compute confidence interval.
- `outx`: set to TRUE to not count pairs of observations tied on `x` as a relevant pair. This results in a Goodman-Kruskal gamma type rank correlation.
- `method`: can take the value conservative, noether or name (see paper Pencina et al. for details).
- `alternative`: a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" (concordance index is greater than 0.5) or "less" (concordance index is less than 0.5). You can specify just the initial letter.
- `na.rm`: TRUE if missing values should be removed.
concordance.index

Value

- `c.index` concordance index estimate.
- `se` standard error of the estimate.
- `lower` lower bound for the confidence interval.
- `upper` upper bound for the confidence interval.
- `p.value` p-value for the statistical test if the estimate if different from 0.5.
- `n` number of samples used for the estimation.
- `data` list of data used to compute the index (x, surv.time and surv.event, or c1).
- `comppairs` number of comparible pairs.

Note

The "direction" of the concordance index (< 0.5 or > 0.5) is the opposite than the `rcorr.cens` function in the `Hmisc` package. So you can easily get the same results than `rcorr.cens` by changing the sign of x.

Author(s)

Benjamin Haibe-Kains, Markus Schroeder

References


See Also

`rcorr.cens, phcpe, coxphCPE`

Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
sex <- sample(0:1, 100, replace=TRUE)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
strat <- sample(1:3, 100, replace=TRUE)
weight <- runif(100, min=0, max=1)
comppairs <- 10
cat("survival prediction:\n")
concordance.index(x=age, surv.time=stime, surv.event=sevent, strat=strat,
weights=weight, method="noether", comppairs=comppairs)
```
cat("binary class prediction:\n")
## is age predictive of sex?
concordance.index(x=age, cl=sex, strat=strat, method="noether")

---

**cvpl**

*Function to compute the CVPL*

**Description**

The function computes the cross-validated partial likelihood (CVPL) for the Cox model.

**Usage**

```r
cvpl(x, surv.time, surv.event, strata, nfold = 1, setseed,
    na.rm = FALSE, verbose = FALSE)
```

**Arguments**

- `x`: data matrix
- `surv.time`: vector of times to event occurrence
- `surv.event`: vector of indicators for event occurrence
- `strata`: stratification variable
- `nfold`: number of folds for the cross-validation
- `setseed`: seed for the random generator
- `na.rm`: TRUE if the missing values should be removed from the data, FALSE otherwise
- `verbose`: verbosity of the function

**Value**

- `cvpl`: mean cross-validated partial likelihood (lower is better)
- `pl`: vector of cross-validated partial likelihoods
- `convergence`: vector of booleans reporting the convergence of the Cox model in each fold
- `n`: number of observations used to estimate the cross-validated partial likelihood

**Author(s)**

Benjamin Haibe-Kains

**References**


**D.index**

Function to compute the D index for a risk prediction, i.e. an estimate of the log hazard ratio comparing two equal-sized prognostic groups. This is a natural measure of separation between two independent survival distributions under the proportional hazards assumption.

### Usage

```r
d.index(x, surv.time, surv.event, weights, strat, alpha = 0.05, method.test = c("logrank", "likelihood"), na.rm = TRUE, ...)
```

### Arguments

- **x**: a vector of risk predictions.
- **surv.time**: a vector of event times.
- **surv.event**: a vector of event occurrence indicators.
- **weights**: weight of each sample.
- **strat**: stratification indicator.
- **alpha**: alpha level to compute confidence interval.
- **method.test**: Statistical test to use in order to compute the p-values related to a D. index, see `summary.coxph` for more details.
- **na.rm**: TRUE if missing values should be removed.
- **...**: additional parameters to be passed to the `coxph` function.

### Details

The D index is computed using the Cox model fitted on the scaled rankits of the risk scores instead of the risk scores themselves. The scaled rankits are the expected standard Normal order statistics scaled by $\kappa = \sqrt{8/\pi}$. See (Royston and Sauerbrei, 2004) for details. Note that the value D reported in (Royston and Sauerbrei, 2004) is given for the test statistic under the null hypothesis $\lambda_1 = \lambda_2$, where $\lambda_1$ and $\lambda_2$ are the log hazard ratios in the two groups.

### Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime == cens)
stime <- pmin(stime, cens)
strat <- sample(1:3, 100, replace=TRUE)
cvpl(x=age, surv.time=stimes, surv.event=sevent, strata=strat,
     nfold=1, setseed=54321)
```
Value

- `d.index`: D index estimate.
- `se`: standard error of the estimate.
- `lower`: lower bound for the confidence interval.
- `upper`: upper bound for the confidence interval.
- `p.value`: p-value for the statistical test if the estimate if different from 0.5.
- `n`: number of samples used for the estimation.
- `coxm`: `coxph.object` fitted on the survival data and z (see below).
- `data`: list of data used to compute the index (x, z, surv.time and surv.event). The item z contains the scaled rankits which are the expected standard Normal order statistics scaled by kappa.

Author(s)

Benjamin Haibe-Kains

References


See Also

`coxph`, `coxph.object`, `normOrder`

Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
strat <- sample(1:3, 100, replace=TRUE)
weight <- runif(100, min=0, max=1)
D.index(x=age, surv.time=stime, surv.event=sevent, weights=weight, strat=strat)
```

dindex.comp  

Function to compare two D indices

Description

This function compares two D indices from their betas and standard errors as computed by a Cox model for instance. The statistical test is a Student t test for dependent samples. The two D indices must be computed using the `D.index` function from the same survival data.
Usage

```r
dindex.comp(dindex1, dindex2)
```

Arguments

- `dindex1` first D index
- `dindex2` second D index

Details

The two D indices must be computed using the `D.index` function from the same samples (and corresponding survival data). The function uses a Student t test for dependent samples.

Value

- `p.value` p-value from the Wilcoxon rank sum test for the comparison `dindex1 > dindex2`
- `dindex1` value of the first D index
- `dindex2` value of the second D index

Author(s)

Benjamin Haibe-Kains

References


See Also

- `D.index`, `t.test`

Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
size <- rexp(100, 1)
stime <- rexp(100)
cens <- runif(100, .5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
d1 <- D.index(x=age, surv.time=stime, surv.event=sevent)
d2 <- D.index(x=size, surv.time=stime, surv.event=sevent)
dindex.comp(d1, d2)
```
dindex.comp.meta  

Function to compare two D indices

Description

This function compares two lists of D indices computed from the same survival data by using the function `d.index`. The statistical test is a Student t test for dependent samples.

Usage

dindex.comp.meta(list.dindex1, list.dindex2, hetero = FALSE)

Arguments

- `list.dindex1`: first list of D indices as returned by the `d.index` function.
- `list.dindex2`: second list of D indices as returned by the `d.index` function.
- `hetero`: if TRUE, a random effect model is used to compute the meta-estimators. Otherwise a fixed effect model is used.

Details

In meta-analysis, we estimate the statistic of interest in several independent datasets. It results a list of estimates such as list of D indices. The two lists of D indices must be computed from the same samples (and corresponding survival data). The function computes a meta-estimator for the correlations between the two scores and uses a Student t test for dependent samples.

Value

- `p.value`: p-value from the Student t test for the comparison `dindex1 > dindex2`.
- `dindex1`: meta-estimator of the first D index.
- `dindex2`: meta-estimator of the second D index.

Author(s)

Benjamin Haibe-Kains

References


See Also

`concordance.index`. 

---

**dindex.comp.meta**

*Function to compare two D indices*

**Description**

This function compares two lists of D indices computed from the same survival data by using the function `d.index`. The statistical test is a Student t test for dependent samples.

**Usage**

dindex.comp.meta(list.dindex1, list.dindex2, hetero = FALSE)

**Arguments**

- `list.dindex1`: first list of D indices as returned by the `d.index` function.
- `list.dindex2`: second list of D indices as returned by the `d.index` function.
- `hetero`: if TRUE, a random effect model is used to compute the meta-estimators. Otherwise a fixed effect model is used.

**Details**

In meta-analysis, we estimate the statistic of interest in several independent datasets. It results a list of estimates such as list of D indices. The two lists of D indices must be computed from the same samples (and corresponding survival data). The function computes a meta-estimator for the correlations between the two scores and uses a Student t test for dependent samples.

**Value**

- `p.value`: p-value from the Student t test for the comparison `dindex1 > dindex2`.
- `dindex1`: meta-estimator of the first D index.
- `dindex2`: meta-estimator of the second D index.

**Author(s)**

Benjamin Haibe-Kains

**References**


**See Also**

`concordance.index`. 

Examples

```r
# first dataset
set.seed(12345)
age <- rnorm(100, 50, 10)
size <- rexp(100, 1)
stime <- rexp(100)
cens <- runif(100, 0, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
d1.1 <- D.index(x=age, surv.time=stime, surv.event=sevent)
d2.1 <- D.index(x=size, surv.time=stime, surv.event=sevent)

# second dataset
set.seed(54321)
age <- rnorm(110, 53, 10)
size <- rexp(110, 1.1)
stime <- rexp(110)
cens <- runif(110, 0, 55, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
d1.2 <- D.index(x=age, surv.time=stime, surv.event=sevent)
d2.2 <- D.index(x=size, surv.time=stime, surv.event=sevent)
dindex.comp.meta(list.dindex=list("dindex.age1"=d1.1, "dindex.age2"=d1.2),
                 list.dindex2=list("dindex.size1"=d2.1, "dindex.size2"=d2.2))
```

---

**fisherz**

*Function to compute Fisher z transformation*

---

**Description**

The function computes the Fisher z transformation useful to calculate the confidence interval of Pearson’s correlation coefficient.

**Usage**

`fisherz(x, inv = FALSE, eps = 1e-16)`

**Arguments**

- **x**
  - Value, e.g. Pearson’s correlation coefficient
- **inv**
  - TRUE for inverse Fisher z transformation, FALSE otherwise
- **eps**
  - Tolerance for extreme cases, i.e.
    
    \[
    \text{latex when inv = FALSE and latex when inv = TRUE}
    \]
Details

The sampling distribution of Pearson’s \textit{latex} is not normally distributed. R. A. Fisher developed a transformation now called “Fisher’s z transformation” that converts Pearson’s \textit{latex} to the normally distributed variable \textit{z}. The formula for the transformation is

\[ \text{latex} \]

Two attributes of the distribution of the \textit{z} statistic: (1) It is normally distributed and (2) it has a known standard error of

\[ \text{latex} \]

where \textit{latex} is the number of samples.

Fisher’s \textit{z} is used for computing confidence intervals on Pearson’s correlation and for confidence intervals on the difference between correlations.

Value

Fisher’s \textit{z} statistic

Author(s)

Benjamin Haibe-Kains

References

R. A. Fisher (1915) "Frequency distribution of the values of the correlation coefficient in samples of an indefinitely large population". \textit{Biometrika}, 10, pages 507–521.

See Also

cor

Examples

```r
set.seed(12345)
x1 <- rnorm(100, 50, 10)
x2 <- runif(100, 5, 2)
c <- cor(x1, x2)
z <- fisherz(x=c, inv=FALSE)
z.se <- 1 / sqrt(100 - 3)
fisherz(z, inv=TRUE)
```
Forest plots enable to display performance estimates of survival models

Description

Draw a forest plot together with a table of text.

Usage

```r
forestplot surv(labeltext, mean, lower, upper, align = NULL,
                is.summary = FALSE, clip = c(-Inf, Inf), xlab = "", zero = 0,
                graphwidth = unit(2, "inches"), col, xlog = FALSE,
                box.size = NULL, x.ticks = NULL, ...)```

Arguments

- `labeltext`: Matrix of strings or NAs for blank spaces
- `mean`: Vector of centers of confidence intervals (or NAs for blank space)
- `lower`: Vector of lower ends of confidence intervals
- `upper`: Vector of upper ends of confidence intervals
- `align`: Vector giving alignment (l,r,c) for columns of table
- `is.summary`: Vector of logicals. Summary lines have bold text and diamond confidence intervals.
- `clip`: Lower and upper limits for clipping confidence intervals to arrows
- `xlab`: x-axis label
- `zero`: x-axis coordinate for zero line
- `graphwidth`: Width of confidence interval graph
- `col`: See `meta.colors`
- `xlog`: If TRUE, x-axis tick marks are exponentiated
- `box.size`: Override the default box size based on precision
- `x.ticks`: Optional user-specified x-axis tick marks. Specify NULL to use the defaults, numeric(0) to omit the x-axis.
- `...`: Not used.

Details

This function is more flexible than `metaplot` and the plot methods for meta-analysis objects, but requires more work by the user.

In particular, it allows for a table of text, and clips confidence intervals to arrows when they exceed specified limits.
Value

None

References

rmeta package, CRAN, Thomas Lumley <tlumley@u.washington.edu>. Functions for simple fixed and random effects meta-analysis for two-sample comparisons and cumulative meta-analyses. Draws standard summary plots, funnel plots, and computes summaries and tests for association and heterogeneity.

See Also

metaplot, forestplot

Examples

```r
require(rmeta)
myspace <- ""
labeltext <- cbind(c("Gene Symbol", "AAA", "BBB", "CCC"), c(rep(myspace, 4)))
bs <- rep(0.5, nrow(labeltext))
r.mean <- c(NA, 0.35, 0.5, 0.65)
r.lower <- c(NA, 0.33, 0.4, 0.6)
r.upper <- c(NA, 0.37, 0.6, 0.7)
forestplot.surv(labeltext=labeltext, mean=r.mean, lower=r.lower, upper=r.upper, zero=0.5,
align=c("1"), graphwidth=unit(2, "inches"), x.ticks=seq(0.3, 0.8, 0.1), xlab=paste("Forestplot Example", myspace),
col=meta.colors(box="royalblue", line="darkblue", zero="darkred"), box.size=bs, clip=c(0.3, 0.8))
```

---

**getsurv2**

*Function to retrieve the survival probabilities at a specific point in time*

Description

The function retrieves the survival probabilities from a survfit object, for a specific point in time.

Usage

```r
getsurv2(sf, time, which.est = c("point", "lower", "upper"))
```

Arguments

- **sf**: survfit object
- **time**: time at which the survival probabilities must be retrieved
- **which.est**: which estimation to be returned? point for the point estimate, lower for the lower bound and upper for the upper bound
Details

The survival probabilities are estimated through the `survfit` function.

Value

vector of survival probabilities

Author(s)

Benjamin Haibe-Kains

See Also

`survfit`

Examples

```r
require(survival)
set.seed(12345)
age <- rnorm(30, 50, 10)
stime <- rexp(30)
cens <- runif(30, 0, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
sf <- survfit(Surv(stime, event) ~ 1)
getsurv2(sf, time = 1)
```

hazard.ratio

*Function to estimate the hazard ratio through Cox regression*

Description

Function to compute the hazard ratio for a risk prediction.

Usage

```r
hazard.ratio(x, surv.time, surv.event, weights, strat, alpha = 0.05, method.test = c("logrank", "likelihood"))
```

Arguments

- `x`: a vector of risk predictions.
- `surv.time`: a vector of event times.
- `surv.event`: a vector of event occurrence indicators.
- `weights`: weight of each sample.
- `strat`: stratification indicator.
- `alpha`: alpha level to compute confidence interval.
method.test    Statistical test to use in order to compute the p-values related to a D. index, see summary.coxph for more details.
na.rm          TRUE if missing values should be removed.
...            additional parameters to be passed to the coxph function.

Details

The hazard ratio is computed using the Cox model.

Value

hazard.ratio    hazard ratio estimate.
coef            coefficient (beta) estimated in the cox regression model.
se              standard error of the coefficient (beta) estimate.
lower           lower bound for the confidence interval.
upper           upper bound for the confidence interval.
p.value         p-value computed using the likelihood ratio test whether the hazard ratio is different from 1.
n              number of samples used for the estimation.
coxm            coxph.object fitted on the survival data and x (see below).
data            list of data used to compute the hazard ratio (x, surv.time and surv.event).

Author(s)

Benjamin Haibe-Kains

References


See Also

coxph, coxph.object

Examples

set.seed(12345)
age <- rnorm(100, 50, 10)
stime <- rexp(100)
cens <- runif(100, .5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
strat <- sample(1:3, 100, replace=TRUE)
weight <- runif(100, min=0, max=1)
hazard.ratio(x=age, surv.time=stime, surv.event=sevent, weights=weight, strat=strat)
Description

This function compares two hazard ratios from their betas and standard errors as computed by a Cox model for instance. The statistical test is a Student t test for dependent samples. The two hazard ratios must be computed from the same survival data.

Usage

hr.comp(hr1, hr2)

Arguments

hr1 first hazard ratio.
hr2 second hazard ratio.

Details

The two hazard ratios must be computed from the same samples (and corresponding survival data). The function uses a Student t test for dependent samples.

Value

p.value p-value from the Student t test for the comparison beta1 > beta2 (equivalently hr1 > hr2)
hr1 value of the first hazard ratio
hr2 value of the second hazard ratio

Author(s)

Benjamin Haibe-Kains

References

Student 1908) "The Probable Error of a Mean", *Biometrika*, 6, 1, pages 1–25.

See Also

coxph, t.test
hr.comp.meta

Function to compare two concordance indices

Description

This function compares two lists of hazard ratios computed from the same survival data by using the function hazard.ratio. The statistical test is a Student t test for dependent samples.

Usage

hr.comp.meta(list.hr1, list.hr2, hetero = FALSE)

Arguments

list.hr1 first list of D indices as returned by the hazard.ratio function.
list.hr2 second list of D indices as returned by the hazard.ratio function.
hetero if TRUE, a random effect model is use to compute the meta-estimators. Otherwise a fixed effect model is used.

Details

In meta-analysis, we estimate the statistic of interest in several independent datasets. It results a list of estimates such as list of hazard ratios. The two lists of hazard ratios must be computed from the same samples (and corresponding survival data). The function computes a meta-estimator for the correlations between the two scores and uses a Student t test for dependent samples.

Value

p.value p-value from the Student t test for the comparison hr1 > hr2.
hr1 meta-estimator of the first D index.
hr2 meta-estimator of the second D index.

Author(s)

Benjamin Haibe-Kains
References


See Also

`concordance.index`

Examples

```r
# first dataset
set.seed(12345)
age <- rnorm(100, 50, 10)
size <- rexp(100, 1)
stime <- rexp(100)
cens <- runif(100, 5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
h1.1 <- hazard.ratio(x = age, surv.time = stime, surv.event = sevent)
h2.1 <- hazard.ratio(x = size, surv.time = stime, surv.event = sevent)

# second dataset
set.seed(54321)
age <- rnorm(110, 53, 10)
size <- rexp(110, 1.1)
stime <- rexp(110)
cens <- runif(110, 55, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
h1.2 <- hazard.ratio(x = age, surv.time = stime, surv.event = sevent)
h2.2 <- hazard.ratio(x = size, surv.time = stime, surv.event = sevent)
hr.comp.meta(list.hr1 = list("hr.age1" = h1.1, "hr.age2" = h1.2),
             list.hr2 = list("hr.size1" = h2.1, "hr.size2" = h2.2))
```

hr.comp2  
*Function to statistically compare two hazard ratios (alternative interface)*

Description

This function compares two hazard ratios from their betas and standard errors as computed by a Cox model for instance. The statistical test is a Student t test for dependent samples. The two hazard ratios must be computed from the same survival data.

Usage

```r
hr.comp2(x1, beta1, se1, x2, beta2, se2, n)
```
Arguments

- `x1` risk score used to estimate the first hazard ratio.
- `beta1` beta estimate for the first hazard ratio.
- `se1` standard error of beta estimate for the first hazard ratio.
- `x2` risk score used to estimate the second hazard ratio.
- `beta2` beta estimate for the second hazard ratio.
- `se2` standard error of beta estimate for the first hazard ratio.
- `n` number of samples from which the hazard ratios were estimated.

Details

The two hazard ratios must be computed from the same samples (and corresponding survival data). The function uses a Student t test for dependent samples.

Value

- `p.value` p-value from the Student t test for the comparison beta1 > beta2 (equivalently hr1 > hr2)
- `hr1` value of the first hazard ratio
- `hr2` value of the second hazard ratio

Author(s)

Benjamin Haibe-Kains

References

Student 1908) "The Probable Error of a Mean", *Biometrika*, 6, 1, pages 1–25.


See Also

coxph, t.test

Examples

```r
require(survival)
set.seed(12345)
age <- as.numeric(rnorm(100, 50, 10) >= 50)
size <- as.numeric(rexp(100, 1) > 1)
stime <- rexp(100)
cens <- runif(100, .5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
coxm1 <- coxph(Surv(stime, sevent) ~ age)
```
iauc.comp

Function to compare two IAUCs through time-dependent ROC curves

Description

This function compares two integrated areas under the curves (IAUC) through the results of time-dependent ROC curves at some points in time. The statistical test is a Wilcoxon rank sum test for dependent samples.

Usage

`iauc.comp(auc1, auc2, time)`

Arguments

- `auc1`: vector of AUCs computed from the first time-dependent ROC curves for some points in time
- `auc2`: vector of AUCs computed from the second time-dependent ROC curves for some points in time
- `time`: vector of points in time for which the AUCs are computed

Details

The two vectors of AUCs must be computed from the same samples (and corresponding survival data) and for the same points in time. The function uses a Wilcoxon rank sum test for dependent samples.

Value

- `p.value`: p-value from the Wilcoxon rank sum test for the comparison `iauc1 > iauc2`
- `iauc1`: value of the IAUC for the first set of time-dependent ROC curves
- `iauc2`: value of the IAUC for the second set of time-dependent ROC curves

Author(s)

Benjamin Haibe-Kains

References


See Also

tdrocc, wilcox.test

Examples

```r
set.seed(12345)
age <- rnorm(30, 50, 10)
size <- rexp(30, 1)
stime <- rexp(30)
cens <- runif(30, .5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
# # time-dependent ROC curves
tt <- unique(sort(stime[sevent == 1]))
# # size
mytdroc1 <- NULL
for (i in 1:length(tt)) {
  rr <- tdrocc(x = size, surv.time = stime, surv.event = event, time = tt[i],
               na.rm = TRUE, verbose = FALSE)
  mytdroc1 <- c(mytdroc1, list(rr))
}
auc1 <- unlist(lapply(mytdroc1, function(x) { return(x$AUC) })))
# # age
mytdroc2 <- NULL
for (i in 1:length(tt)) {
  rr <- tdrocc(x = age, surv.time = stime, surv.event = event, time = tt[i],
               na.rm = TRUE, verbose = FALSE)
  mytdroc2 <- c(mytdroc2, list(rr))
}
auc2 <- unlist(lapply(mytdroc2, function(x) { return(x$AUC) })))
auc.comp(auc1 = auc1, auc2 = auc2, time = tt)
```

---

**ibsc.comp**

*Function to compare two IBSCs*

**Description**

This function compares two integrated Briers scores (IBSC) through the estimation of the Brier scores (BSC) at some points in time. The statistical test is a Wilcoxon rank sum test for dependent samples.

**Usage**

```r
ibsc.comp(bsc1, bsc2, time)
```
Arguments

bsc1  vector of BSCs computed from the first predicted probabilities for some points in time
bsc2  vector of BSCs computed from the second predicted probabilities for some points in time
time vector of points in time for which the BSCs are computed

Details

The two vectors of BSCs must be computed from the same samples (and corresponding survival data) and for the same points in time. The function uses a Wilcoxon rank sum test for dependent samples.

Value

p.value p-value from the Wilcoxon rank sum test for the comparison bsc1 < bsc2
ibsc1 value of the IBSC for the first set of BSCs
ibsc2 value of the IBSC for the second set of BSCs

Author(s)

Benjamin Haibe-Kains

References


See Also

sbrier.score2proba, sbrier

Examples

```r
set.seed(12345)
age <- rnorm(30, 50, 10)
size <- rexp(30, 1)
stime <- rexp(30)
cens <- runif(30,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
#Brier scores
##size
dd <- data.frame("time"=stime, "event"=sevent, "score"=size)
bsc1 <- sbrier.score2proba(data.tr=dd, data.ts=dd, method="cox")
##size
```
dd <- data.frame("time"=stime, "event"=sevent, "score"=age)
bsc2 <- sbrier.score2proba(data.tr=dd, data.ts=dd, method="cox")
if(!all(bsc1$time == bsc2$time)) {
  stop("the two vector of BSCs must be computed for the same points in time!")
}
ibsc.comp(bsc1=bsc1$bsc, bsc2=bsc2$bsc, time=bsc1$time)

---

**km.coxph.plot**

*Function to plot several Kaplan-Meier survival curves*

**Description**

Function to plot several Kaplan-Meier survival curves with number of individuals at risk at some time points.

**Usage**

```r
km.coxph.plot(formula.s, data.s, weight.s, x.label, y.label, main.title, sub.title, leg.text, leg.pos)
```

**Arguments**

- `formula.s` formula composed of a Surv object and a strata variable (i.e. stratification).
- `data.s` data frame composed of the variables used in the formula.
- `weight.s` vector of weights of length nrow(data.s).
- `x.label` label for the y-axis.
- `y.label` label for the x-axis.
- `main.title` main title at the top of the plot.
- `sub.title` subtitle at the bottom of the plot.
- `leg.text` text in the legend.
- `leg.pos` the location may also be specified by setting 'x' to a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". This places the legend on the inside of the plot frame at the given location.
- `leg.bty` the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
- `leg.inset` inset distance from the margins as a fraction of the plot region. Default value is 0.05.
- `o.text` plot the logrank p-value.
- `v.line` x coordinate(s) for vertical line(s).
- `h.line` y coordinate(s) for horizontal line(s).
- `col` vector of colors for the different survival curves.
- `lty` vector of line types for the different survival curves
- `lwd` vector of line widths for the different survival curves.
show.n.risk if TRUE, show the numbers of samples at risk for each time step.
n.risk.step vector specifying the time to be the steps for displaying the number of individuals at risk.
n.risk.cex size of the number of individuals at risk. Default value is 0.85.
verbose verbosity level (TRUE or FALSE). Default value is TRUE.
... additional parameters to be passed to the plot function.

Details

The original version of this function was kindly provided by Dr Christos Hatzis (January, 17th 2006).

Author(s)

Christos Hatzis, Benjamin Haibe-Kains

See Also

survfit, coxph

Examples

set.seed(12345)
stime <- rexp(100) * 10
cens <- runif(100,.5,2) * 10
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
strat <- sample(1:3, 100, replace=TRUE)
dd <- data.frame("surv.time"=stime, "surv.event"=sevent, "strat"=strat)
ddweights <- array(1, dim=nrow(dd))

km.coxph.plot(formula.s=Surv(surv.time, surv.event) ~ strat, data.s=dd, weight.s=ddweights, x.label="Time (years)", y.label="Probability of survival", main.title="", leg.text=paste(c("Low", "Intermediate", "High"), " ", sep=""), leg.pos="topright", leg.inset=0, .col=c("darkblue", "darkgreen", "darkred"), lty=c(1,1,1), show.n.risk=TRUE, n.risk.step=2, n.risk.cex=0.85, verbose=FALSE)
Arguments

- **surv.time**: vector of times to event occurrence
- **surv.event**: vector of indicators for event occurrence
- **pred**: linear predictors computed using the Cox model
- **strata**: stratification variable
- **na.rm**: TRUE if the missing values should be removed from the data, FALSE otherwise
- **verbose**: verbosity of the function

Value

vector of two elements: `logpl` and `event` for the estimation of the log partial likelihood and the number of events, respectively

Author(s)

Benjamin Haibe-Kains

References


See Also

- `coxph`, `cvpl`

Examples

```r
require(survival)
set.seed(12345)
age <- rnorm(100, 50, 10)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
dd <- data.frame("stime"=stime, "sevent"=sevent, "age"=age)

# Cox model
coxm <- coxph(Surv(stime, seven) ~ age, data=dd)

# Log partial likelihood of the null model
logpl(pred=rep(0, nrow(dd)), surv.time=stime, surv.event=sevent)

# Log partial likelihood of the Cox model
logpl(pred=predict(object=coxm, newdata=dd), surv.time=stime, surv.event=sevent)

# Equivalent to
coxm$loglik
```
Description

This dataset contains a subset of the gene expression, annotations and clinical data from the MAINZ datasets (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008

Format

ExpressionSet with 7 features and 200 samples, containing:

- `exprs(mainz7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a technology (single-channel, oligonucleotides).
- `fData(mainz7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a.
- `pData(mainz7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(mainz7g)`: MIAME object containing information about the dataset.
- `annotation(mainz7g)`: Name of the affy chip.

Details

This dataset represents a subset of the study published by Schmidt et al. 2008. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

Source

mainz:


References

Examples

```R
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
eexprs(mainz7g)[,1:5]
## show the first 6 rows of the phenotype data
head(pData(mainz7g))
## show first 20 feature names
featureNames(mainz7g)
## show the experiment data summary
experimentData(mainz7g)
## show the used platform
annotation(mainz7g)
## show the abstract for this dataset
abstract(mainz7g)
```

---

**metaplot.surv**

*Meta-analysis plot (forest plot)*

Description

Plot confidence intervals with boxes indicating the sample size/precision and optionally a diamond indicating a summary confidence interval. This function is usually called by `plot` methods for meta-analysis objects. Additional, you can specify your own lower and upper boarder from the confidence interval.

Usage

```R
metaplot.surv(mn, se=NULL, lower=NULL, upper=NULL, nn=NULL, labels=NULL, conf.level = .95, xlab = "", ylab = "", xlim = NULL, summn = NULL, sumse = NULL, sumlower = NULL, sumupper = NULL, sumnn = NULL, summlabel = "Summary", logeffect = FALSE, lwd = 2, boxsize = 1, zero = as.numeric(logeffect), colors, xaxt="s", logticks=TRUE, ... )
```

Arguments

- `mn` point estimates from studies
- `se` standard errors of `mn`
- `lower` Vector of lower ends of confidence intervals
- `upper` Vector of upper ends of confidence intervals
- `nn` precision: box area is proportional to this. `1/se^2` is the default
- `labels` labels for each interval
- `conf.level` Confidence level for confidence intervals
Function to compute the concordance index for survival or binary class prediction

Function to compute the minimum redundancy - maximum relevance (mRMR) ranking for a risk prediction or a binary classification task based on the concordance index. The mRMR feature selection has been adapted to use the concordance index to estimate the correlation between a variable and the output (binary or survival) data.
Usage

mrmr.cindex(x, surv.time, surv.event, cl, weights, commpairs=10, strat, alpha = 0.05, outx = TRUE, method = "conservative")

Arguments

- **x**: a vector of risk predictions.
- **surv.time**: a vector of event times.
- **surv.event**: a vector of event occurrence indicators.
- **cl**: a vector of binary class indicators.
- **weights**: weight of each sample.
- **commpairs**: threshold for comparable patients.
- **strat**: stratification indicator.
- **alpha**: alpha level to compute confidence interval.
- **outx**: set to TRUE to not count pairs of observations tied on x as a relevant pair. This results in a Goodman-Kruskal gamma type rank correlation.
- **method**: can take the value conservative, noether or name (see paper Pencina et al. for details).
- **alternative**: a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" (concordance index is greater than 0.5) or "less" (concordance index is less than 0.5). You can specify just the initial letter.
- **na.rm**: TRUE if missing values should be removed.

Value

A mRMR ranking

Note

The "direction" of the concordance index (< 0.5 or > 0.5) is the opposite than the rcorr.cens function in the Hmisc package. So you can easily get the same results than rcorr.cens by changing the sign of x.

Author(s)

Benjamin Haibe-Kains, Markus Schroeder

References


Function to compute the concordance index for survival or binary class prediction

Description

Function to compute the minimum redundancy - maximum relevance (mRMR) ranking for a risk prediction or a binary classification task based on the concordance index. The mRMR feature selection has been adapted to use the concordance index to estimate the correlation between a variable and the output (binary or survival) data.

Usage

mrmr.cindex.ensemble(x, surv.time, surv.event, cl, weights, comppairs=10, strat, alpha = 0.05, outx = FALSE)

Arguments

x          a vector of risk predictions.
surv.time  a vector of event times.
surv.event  a vector of event occurrence indicators.
cl          a vector of binary class indicators.
weights     weight of each sample.
comppairs   threshold for comparable patients.
strat       stratification indicator.
alpha       alpha level to compute confidence interval.
outx        set to TRUE to not count pairs of observations tied on x as a relevant pair. This results in a Goodman-Kruskal gamma type rank correlation.
method can take the value conservative, noether or name (see paper Pencina et al. for details).

alternative a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" (concordance index is greater than 0.5) or "less" (concordance index is less than 0.5). You can specify just the initial letter.

maxparents maximum number of candidate variables to be added in the ranking solutions tree.

maxnsol maximum number of ranking solutions to be considered.

nboot number of bootstraps to compute standard error of a ranking solution.

na.rm TRUE if missing values should be removed.

Value

A mRMR ranking

Note

The "direction" of the concordance index (< 0.5 or > 0.5) is the opposite than the rcorr.cens function in the Hmisc package. So you can easily get the same results than rcorr.cens by changing the sign of x.

Author(s)

Benjamin Haibe-Kains, Markus Schroeder

References


See Also

rcorr.cens, phcpe, coxphCPE
**Description**

This dataset contains a subset of the gene expression, annotations and clinical data from the NKI datasets (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008.

**Format**

ExpressionSet with 7 features and 337 samples, containing:

- `exprs(nki7g)`: Matrix containing gene expressions as measured by Agilent technology (dual-channel, oligonucleotides).
- `fdata(nki7g)`: AnnotatedDataFrame containing annotations of Agilent microarray platform.
- `pdata(nki7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentaldata(nki7g)`: MIAME object containing information about the dataset.
- `annotation(nki7g)`: Name of the agilent chip.

**Details**

This dataset represents a subset of the study published by van’t Veer et al. 2002 and van de Vijver et al. 2002. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

**Source**

*nki:*


**References**


**Subset of NKI dataset containing gene expression, annotations and clinical data.**
Examples

```r
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
exprs(nki7g)[,1:5]
## show the first 6 rows of the phenotype data
head(pData(nki7g))
## show first 20 feature names
featureNames(nki7g)
## show the experiment data summary
experimentData(nki7g)
## show the used platform
annotation(nki7g)
## show the abstract for this dataset
abstract(nki7g)
```

---

**no.at.risk**  
*Function to compute the number of individuals at risk*

---

**Description**

Function to compute the number of individuals at risk at certain time points, as used in the Kaplan-Meier estimator for instance, depending on stratification.

**Usage**

```r
no.at.risk(formula.s, data.s, sub.s = "all", t.step, t.end)
```

**Arguments**

- `formula.s`  
  formula composed of a `Surv` object and a strata variable (i.e. stratification).
- `data.s`  
  data frame composed of the variables used in the formula.
- `sub.s`  
  vector of booleans specifying if only a subset of the data should be considered.
- `t.step`  
  time step at which the number of individuals at risk is computed.
- `t.end`  
  maximum time to be considered.

**Details**

The original version of this function was kindly provided by Dr Christos Hatzis (January, 17th 2006).

**Value**

number of individuals at risk at each time step specified in `t.step` up to `t.end`. 
Function to compute the BSCs from a risk score, for all the times of event occurrence

The function computes all the Brier scores (BSC) and the corresponding integrated Brier score (IBSC) from a risk score, for all the times of event occurrence. The risk score is first transformed in survival probabilities using either a Cox model or the product-limit estimator.

Usage

sbrier.score2proba(data.tr, data.ts, method = c("cox", "prodlim"))

Arguments

data.tr the data frame for the training set. This data frame must contain three columns for the times, the event occurrence and the risk score. These columns are called "time", "event" and "score" respectively.
data.ts the data frame for the test set. This data frame must contain three columns for the times, the event occurrence and the risk score. These columns are called "time", "event" and "score" respectively.
method method for survival probabilities estimation using either a Cox model or the product-limit estimator.
Value

- `time` vector of points in time
- `bsc` vector of Brier scores (BSC) at one points in time
- `bsc.integrated` value of the integrated Brier score (IBSC)

Author(s)

Benjamin Haibe-Kains

References


See Also

- `sbrier`, `coxph`, `prodlim`

Examples

```r
set.seed(12345)
age <- rnorm(30, 50, 10)
stime <- rexp(30)
cens <- runif(30, 0.5, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
dd <- data.frame("time"=stime, "event"=sevent, "score"=age)

#Cox's model
sbrier.score2proba(data.tr=dd, data.ts=dd, method="cox")

#product-limit estimator
sbrier.score2proba(data.tr=dd, data.ts=dd, method="prodlim")
```
**score2proba**

*Function to compute the survival probabilities from a risk score*

**Description**

The function uses either a Cox model or the product-limit estimator to compute the survival probabilities from a risk score for a specific point in time.

**Usage**

```r
score2proba(data.tr, score, yr, method = c("cox", "prodlim"),
             conf.int = 0.95, which.est= c("point", "lower", "upper"))
```

**Arguments**

- `data.tr` - the data frame for the training set. This data frame must contain three columns for the times, the event occurrence and the risk score. These columns are called "time", "event" and "score" respectively.
- `score` - risk score for the test set
- `yr` - a point in time for which the survival probabilities must be computed
- `method` - method for survival probabilities estimation, either `cox` or `prodlim` for the Cox model or the product-limit estimator, respectively
- `conf.int` - value in [0,1]. Default at 0.95
- `which.est` - which estimation to be returned? point for the point estimate, lower for the lower bound and upper for the upper bound

**Value**

vector of predicted survival probabilities

**Author(s)**

Benjamin Haibe-Kains

**References**


**See Also**

coxph, prodlim
Examples

```r
set.seed(12345)
age <- rnorm(30, 50, 10)
stime <- rexp(30)
cens <- runif(30,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
dd <- data.frame("time"=stime, "event"=sevent, "score"=age)

# Cox's model
score2proba(data.tr=dd, score=dd$score, yr=1, method="cox")
# product-limit estimator
score2proba(data.tr=dd, score=dd$score, yr=1, method="prodlim")
```

---

**td.sens.spec**  
*Function to compute sensitivity and specificity for a binary classification of survival data*

**Description**

The function is a wrapper for the `survivalROC.C` function in order to compute sensitivity and specificity for a binary classification of survival data.

**Usage**

```r
td.sens.spec(cl, surv.time, surv.event, time, span = 0, sampling = FALSE, na.rm = FALSE, ...)
```

**Arguments**

- `cl`: vector of binary classes.
- `surv.time`: vector of times to event occurrence.
- `surv.event`: vector of event occurrence indicators.
- `time`: time point for sensitivity and specificity estimations.
- `span`: Span for the NNE. Default value is 0.
- `sampling`: jackknife procedure to estimate the standard error of sensitivity and specificity estimations.
- `na.rm`: TRUE if the missing values should be removed from the data, FALSE otherwise.
- `...`: additional arguments to be passed to the `survivalROC` function.

**Details**

Only NNE method is used to estimate sensitivity and specificity (see `survivalROC.C`). The standard error for sensitivity and specificity is estimated through jackknife procedure (see `jackknife`).
tdrocc

Value

- sens: sensitivity estimate
- sens.se: standard error for sensitivity estimate
- spec: specificity estimate
- spec.se: standard error for specificity estimate

Author(s)

Benjamin Haibe-Kains

References


See Also

survivalROC

Examples

```r
set.seed(12345)
gender <- sample(c(0,1), 100, replace=TRUE)
stime <- rexp(100)
cens <- runif(100,.5,2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
mysenspec <- td.sens.spec(cl=gender, surv.time=stime, surv.event=sevent,
                        time=1, span=0, na.rm=FALSE)
```

---

Function to compute time-dependent ROC curves

Description

The function is a wrapper for the survivalROC function in order to compute the time-dependent ROC curves.

Usage

```r
tdrocc(x, surv.time, surv.event, surv.entry = NULL, time, cutpts = NA,
na.rm = FALSE, verbose = FALSE, span = 0, lambda = 0, ...)
```
Arguments

- **x**: vector of risk scores.
- **surv.time**: vector of times to event occurrence.
- **surv.event**: vector of event occurrence indicators.
- **surv.entry**: entry time for the subjects.
- **time**: time point for the ROC curve.
- **cutpts**: cut points for the risk score.
- **na.rm**: TRUE if the missing values should be removed from the data, FALSE otherwise.
- **verbose**: verbosity of the function.
- **span**: Span for the NNE, need either lambda or span for NNE.
- **lambda**: smoothing parameter for NNE.
- **...**: additional arguments to be passed to the `survivalROC` function.

Value

- **spec**: specificity estimates
- **sens**: sensitivity estimates
- **rule**: rule to compute the predictions at each cutoff
- **cuts**: cutoffs
- **time**: time point at which the time-dependent ROC is computed
- **survival**: overall survival at the time point
- **AUC**: Area Under the Curve (AUC) of the time-dependent ROC curve
- **data**: survival data and risk score used to compute the time-dependent ROC curve

Author(s)

Benjamin Haibe-Kains

References


See Also

`survivalROC`
Examples

```r
set.seed(12345)
age <- rnorm(100, 50, 10)
stime <- rexp(100)
cens <- runif(100, 0, 2)
sevent <- as.numeric(stime <= cens)
stime <- pmin(stime, cens)
tdroc <- tdrocc(x=age, surv.time=stime, surv.event=sevent, time=1,
               na.rm=TRUE, verbose=FALSE)
# plot the time-dependent ROC curve
plot(x=1-tdroc$spec, y=tdroc$sens, type="l", xlab="1 - specificity",
     ylab="sensitivity", xlim=c(0, 1), ylim=c(0, 1))
lines(x=c(0,1), y=c(0,1), lty=3, col="red")
```

Function to test the heterogeneity of set of probabilities

description

The function tests whether a set of p-values are heterogeneous.

Usage

test.hetero.est(x, x.se, na.rm = FALSE)

Arguments

- `x`: vector of estimates
- `x.se`: vector of standard errors of the corresponding estimates
- `na.rm`: TRUE if the missing values should be removed from the data, FALSE otherwise

Details

The heterogeneity test is known to be very conservative. Consider a p-value < 0.1 as significant.

Value

- `Q`: Q statistic
- `p.value`: p-value of the heterogeneity test

Author(s)

Benjamin Haibe-Kains

References

See Also

combine.test

Examples

```r
set.seed(12345)
x1 <- rnorm(100, 50, 10) + rnorm(100, 0, 2)
m1 <- mean(x1)
se1 <- sqrt(var(x1))
x2 <- rnorm(100, 75, 15) + rnorm(100, 0, 5)
m2 <- mean(x2)
se2 <- sqrt(var(x2))

test.hetero.est(x=c(m1, m2), x.se=c(se1, se2))
```

---

### Description

The function tests whether a set of p-values are heterogeneous.

### Usage

```r
test.hetero.test(p, weight, na.rm = FALSE)
```

### Arguments

- `p` vector of p-values
- `weight` vector of weights (e.g. sample size of each study)
- `na.rm` TRUE if the missing values should be removed from the data, FALSE otherwise

### Details

The p-values should be one-sided and computed from the same null hypothesis.

### Value

- `Q` Q statistic
- `p.value` p-value of the heterogeneity test

### Author(s)

Benjamin Haibe-Kains
References


See Also

combine.test

Examples

```r
p <- c(0.01, 0.13, 0.07, 0.2)
w <- c(100, 50, 200, 30)

# with equal weights
test.hetero.test(p=p)
# with p-values weighted by the sample size of the studies
test.hetero.test(p=p, weight=w)
```

---

**transbig7g**

*Subset of the TRANSBIG dataset containing gene expression, annotations and clinical data.*

Description

This dataset contains a subset of the gene expression, annotations and clinical data from the TRANSBIG dataset (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008

Format

ExpressionSet with 7 features and 198 samples, containing:

- `exprs(transbig7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a technology (single-channel, oligonucleotides).
- `fData(transbig7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a.
- `pData(transbig7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(transbig7g)`: MIAME object containing information about the dataset.
- `annotation(transbig7g)`: Name of the affy chip.
Details

This dataset represents a subset of the study published by Desmedt et al. 2007. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

Source

transbig:


References


Examples

```r
## load Biobase package
library(BiocBase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
exprs(transbig7g)[ ,1:5]
## show the first 6 rows of the phenotype data
head(pData(transbig7g))
## show first 20 feature names
featureNames(transbig7g)
## show the experiment data summary
experimentData(transbig7g)
## show the used platform
annotation(transbig7g)
## show the abstract for this dataset
abstract(transbig7g)
```

Description

This dataset contains a subset of the gene expression, annotations and clinical data from the UNT datasets (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008
**Format**

ExpressionSet with 7 features and 137 samples, containing:

- `exprs(unt7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a and hgu133b technology (single-channel, oligonucleotides).
- `fData(unt7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a and hgu133b.
- `pData(unt7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(unt7g)`: MIAME object containing information about the dataset.
- `annotation(unt7g)`: Name of the affy chip.

**Details**

This dataset represents a subset of the study published by Sotiriou et al. 2006. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

**Source**

unt:


**References**


**Examples**

```r
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
eprs(unt7g)[,1:5]
## show the first 6 rows of the phenotype data
head(pData(unt7g))
## show first 20 feature names
featureNames(unt7g)
## show the experiment data summary
```
Subset of UPP dataset containing gene expression, annotations and clinical data.

Description

This dataset contains a subset of the gene expression, annotations and clinical data from the UPP datasets (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008.

Format

ExpressionSet with 7 features and 251 samples, containing:

- `exprs(upp7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a and hgu133b technology (single-channel, oligonucleotides).
- ` pData(upp7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a and hgu133b.
- ` pData(upp7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(upp7g)`: MIAME object containing information about the dataset.
- `annotation(upp7g)`: Name of the affy chip.

Details

This dataset represents a subset of the study published by Miller et al. 2005. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.

Source

upp:

References

Examples

```r
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
exprs(upp7g)[ ,1:5]
## show the first 6 rows of the phenotype data
head(pData(upp7g))
## show first 20 feature names
featureNames(upp7g)
## show the experiment data summary
experimentData(upp7g)
## show the used platform
annotation(upp7g)
## show the abstract for this dataset
abstract(upp7g)
```

---

**vdx7g**

Subset of VDX dataset containing gene expression, annotations and clinical data.

---

**Description**

This dataset contains a subset of the gene expression, annotations and clinical data from the VDX datasets (see section details). The subset contains the seven genes introduced by Desmedt et al. 2008.

**Format**

ExpressionSet with 7 features and 344 samples, containing:

- `exprs(vdx7g)`: Matrix containing gene expressions as measured by Affymetrix hgu133a technology (single-channel, oligonucleotides).
- `fData(vdx7g)`: AnnotatedDataFrame containing annotations of Affy microarray platform hgu133a.
- `pData(vdx7g)`: AnnotatedDataFrame containing Clinical information of the breast cancer patients whose tumors were hybridized.
- `experimentalData(vdx7g)`: MIAME object containing information about the dataset.
- `annotation(vdx7g)`: Name of the affy chip.

**Details**

This dataset represents a subset of the study published by Wang et al. 2005 and Minn et al. 2007. The subset contains the genes AURKA (also known as STK6, STK7, or STK15), PLAU (also known as UPA), STAT1, VEGF, CASP3, ESR1, and ERBB2, as introduced by Desmedt et al. 2008. The seven genes represent the proliferation, tumor invasion/metastasis, immune response, angiogenesis, apoptosis phenotypes, and the ER and HER2 signaling, respectively.
Source

**vdx:**


References


Examples

```r
## load Biobase package
library(Biobase)
## load the dataset
data(breastCancerData)
## show the first 5 columns of the expression data
exprs(vdx7g)[,1:5]
## show the first 6 rows of the phenotype data
head pData(vdx7g))
## show first 20 feature names
featureNames(vdx7g)
## show the experiment data summary
experimentData(vdx7g)
## show the used platform
annotation(vdx7g)
## show the abstract for this dataset
abstract(vdx7g)
```
Index

*Topic **datasets**
  - breastCancerData, 4
  - mainz7g, 35
  - nki7g, 41
  - transbig7g, 51
  - unt7g, 52
  - upp7g, 54
  - vdx7g, 55

*Topic **hplot**
  - forestplot.surv, 21
  - metaplot.surv, 36

*Topic **htest**
  - cindex.comp, 7
  - cindex.comp.meta, 8
  - dindex.comp, 16
  - dindex.comp.meta, 18
  - hr.comp, 25
  - hr.comp.meta, 26
  - hr.comp2, 27
  - iauc.comp, 29
  - ibsc.comp, 30
  - survcomp-package, 2
  - test.hetero.est, 49
  - test.hetero.test, 50

*Topic **survival**
  - censor.time, 6
  - cindex.comp, 7
  - cindex.comp.meta, 8
  - concordance.index, 12
  - cvpl, 14
  - D.index, 15
  - dindex.comp, 16
  - dindex.comp.meta, 18
  - getsurv2, 22
  - hazard.ratio, 23
  - hr.comp, 25
  - hr.comp.meta, 26
  - hr.comp2, 27
  - iauc.comp, 29
  - ibsc.comp, 30
  - km.coxph.plot, 32
  - logpl, 33
  - mrrm.cindex, 37
  - mrrm.cindex.ensemble, 39
  - no.at.risk, 42
  - sbrier.score2proba, 43
  - score2proba, 45
  - survcomp-package, 2
  - td.sens.spec, 46
  - tdrocc, 47

*Topic **univar**
  - combine.est, 10
  - combine.test, 11
  - concordance.index, 12
  - D.index, 15
  - fisherz, 19
  - hazard.ratio, 23
  - km.coxph.plot, 32
  - mrrm.cindex, 37
  - mrrm.cindex.ensemble, 39
  - no.at.risk, 42
  - sbrier.score2proba, 43

  breastCancerData, 4
  - censor.time, 6
  - cindex.comp, 7
  - cindex.comp.meta, 8
  - combine.est, 10
  - combine.test, 11
  - concordance.index, 8, 9, 12, 18, 27
  - cor, 20
  - coxph, 15, 16, 24, 25, 28, 33, 34, 44, 45
  - coxph.object, 16, 24
  - coxphCPE, 13, 39, 40
  - cvpl, 14, 34

  D.index, 15, 16, 17
  - dindex.comp, 16
<table>
<thead>
<tr>
<th>Function/Package</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>dindex.comp.meta</td>
<td>18</td>
</tr>
<tr>
<td>fisherz</td>
<td>19</td>
</tr>
<tr>
<td>forestplot.surv</td>
<td>21, 37</td>
</tr>
<tr>
<td>getsurv2</td>
<td>22</td>
</tr>
<tr>
<td>hazard.ratio</td>
<td>23</td>
</tr>
<tr>
<td>hr.comp</td>
<td>25</td>
</tr>
<tr>
<td>hr.comp.meta</td>
<td>26</td>
</tr>
<tr>
<td>hr.comp2</td>
<td>27</td>
</tr>
<tr>
<td>iauc.comp</td>
<td>29</td>
</tr>
<tr>
<td>ibsc.comp</td>
<td>30</td>
</tr>
<tr>
<td>jackknife</td>
<td>46</td>
</tr>
<tr>
<td>km.coxph.plot</td>
<td>32, 43</td>
</tr>
<tr>
<td>logpl</td>
<td>15, 33</td>
</tr>
<tr>
<td>mainz7g</td>
<td>35</td>
</tr>
<tr>
<td>meta.colors</td>
<td>21, 37</td>
</tr>
<tr>
<td>metaplot</td>
<td>21</td>
</tr>
<tr>
<td>metaplot.surv</td>
<td>36</td>
</tr>
<tr>
<td>mrmr.cindex</td>
<td>37</td>
</tr>
<tr>
<td>mrmr.cindex.ensemble</td>
<td>39</td>
</tr>
<tr>
<td>nki7g</td>
<td>41</td>
</tr>
<tr>
<td>no.nat.risk</td>
<td>42</td>
</tr>
<tr>
<td>normOrder</td>
<td>16</td>
</tr>
<tr>
<td>phcpe</td>
<td>13, 39, 40</td>
</tr>
<tr>
<td>plot</td>
<td>33</td>
</tr>
<tr>
<td>prodlim</td>
<td>44, 45</td>
</tr>
<tr>
<td>rcorr.cens</td>
<td>13, 38–40</td>
</tr>
<tr>
<td>sbrier</td>
<td>31, 44</td>
</tr>
<tr>
<td>sbrier.score2proba</td>
<td>31, 43</td>
</tr>
<tr>
<td>score2proba</td>
<td>45</td>
</tr>
<tr>
<td>summary.coxph</td>
<td>15, 24</td>
</tr>
<tr>
<td>survcomp(survcomp-package)</td>
<td>2</td>
</tr>
<tr>
<td>survcomp-package</td>
<td>2</td>
</tr>
<tr>
<td>survfit</td>
<td>23, 33, 43</td>
</tr>
<tr>
<td>survivalROC</td>
<td>46–48</td>
</tr>
<tr>
<td>survivalROC.C</td>
<td>46</td>
</tr>
<tr>
<td>t.test</td>
<td>17, 25, 28</td>
</tr>
<tr>
<td>td.sens.spec</td>
<td>46</td>
</tr>
<tr>
<td>tdrocc</td>
<td>30, 47</td>
</tr>
<tr>
<td>test.hetero.est</td>
<td>49</td>
</tr>
<tr>
<td>test.hetero.test</td>
<td>50</td>
</tr>
<tr>
<td>transbig7g</td>
<td>51</td>
</tr>
<tr>
<td>unt7g</td>
<td>52</td>
</tr>
<tr>
<td>upp7g</td>
<td>54</td>
</tr>
<tr>
<td>vdx7g</td>
<td>55</td>
</tr>
<tr>
<td>wilcox.test</td>
<td>30</td>
</tr>
</tbody>
</table>