Package ‘riskRegression’

February 8, 2020

Type Package

Title Risk Regression Models and Prediction Scores for Survival Analysis with Competing Risks

Version 2020.02.05

Depends R (>= 3.5.0),

Imports data.table (>= 1.12.2), ggplot2 (>= 3.1.0), prodlim (>= 2018.4.18), doParallel, stats, graphics, survival (>= 2.44.1), lava (>= 1.6.5), cmprsk, plotrix, timereg (>= 1.9.3), foreach, ranger, parallel, Rcpp, lattice, Publish, rms (>= 5.1.3)

LinkingTo Rcpp, RcppArmadillo

Suggests boot, smcfc, CoxBoost, ipw, SuperLearner, gbm, glmnetUtils, flexsurv, mets, party, pec, penalized, pROC, randomForest, randomForestSRC, rbenchmark, rpart, testthat, R.rsp

Maintainer Thomas Alexander Gerds <tag@biostat.ku.dk>

Description Implementation of the following methods for event history analysis.
Risk regression models for survival endpoints also in the presence of competing risks are fitted using binomial regression based on a time sequence of binary event status variables. A formula interface for the Fine-Gray regression model and an interface for the combination of cause-specific Cox regression models. A toolbox for assessing and comparing performance of risk predictions (risk markers and risk prediction models). Prediction performance is measured by the Brier score and the area under the ROC curve for binary possibly time-dependent outcome. Inverse probability of censoring weighting and pseudo values are used to deal with right censored data. Lists of risk markers and lists of risk models are assessed simultaneously. Cross-validation repeatedly splits the data, trains the risk prediction models on one part of each split and then summarizes and compares the performance across splits.

License GPL (>= 2)

RoxygenNote 7.0.2

VignetteBuilder R.rsp

NeedsCompilation yes
Author Thomas Alexander Gerds [aut, cre],
Paul Blanche [ctb],
Rikke Mortensen [ctb],
Nikolaj Tollenaar [ctb],
Ulla Brasch Mogensen [ctb],
Brice Ozenne [aut] (<https://orcid.org/0000-0001-9694-2956>)

Repository CRAN

Date/Publication 2020-02-08 16:30:02 UTC

R topics documented:

- as.data.table.ate ..................................................... 4
- as.data.table.influenceTest ....................................... 5
- as.data.table.predictCox .......................................... 5
- as.data.table.predictCSC ......................................... 6
- ate ................................................................. 6
- autoplot.ate ....................................................... 12
- autoplot.predictCox .............................................. 13
- autoplot.predictCSC .............................................. 15
- autoplot.Score ................................................... 16
- boot2pvalue ....................................................... 18
- boxplot.Score .................................................... 19
- calcSeCox .......................................................... 21
- calcSeCSC .......................................................... 23
- Cforest .............................................................. 25
- coef.CauseSpecificCox ........................................... 26
- coef.riskRegression .............................................. 26
- colCenter_cpp ...................................................... 27
- colCumProd .......................................................... 27
- colCumSum ........................................................... 28
- colMultiply_cpp ..................................................... 29
- colScale_cpp ........................................................ 29
- colSumsCrossprod .................................................. 30
- confBandCox ........................................................ 31
- confint.ate .......................................................... 31
- confint.influenceTest ............................................. 33
- confint.predictCox ................................................ 34
- confint.predictCSC ............................................... 36
- coxBaseEstimator ................................................... 38
- coxCenter ............................................................ 38
- coxFormula .......................................................... 39
- coxLP ................................................................. 40
- coxModelFrame ....................................................... 40
- coxN ................................................................. 41
- coxSpecial ........................................................... 42
- coxStrata ............................................................ 43
- coxStrataLevel ....................................................... 44
R topics documented:

- `coxVarCov` ................................................. 44
- `coxVariableName` ....................................... 45
- `CSC` ..................................................... 46
- `Ctree` .................................................... 49
- `discreteRoot` ............................................ 50
- `FGR` ....................................................... 50
- `getSplitMethod` .......................................... 52
- `iidCox` ................................................... 53
- `influenceTest` ........................................... 56
- `IPA` ....................................................... 59
- `ipcw` ..................................................... 61
- `Melanoma` ................................................ 64
- `model.matrix.cph` ....................................... 65
- `model.matrix.phreg` .................................... 65
- `Paquid` ................................................... 66
- `penalizedS3` ............................................. 67
- `plot.riskRegression` .................................... 68
- `plotAUC` .................................................. 69
- `plotBrier` ............................................... 71
- `plotCalibration` ......................................... 72
- `plotEffects` ............................................. 75
- `plotPredictRisk` ........................................ 77
- `plotRisk` ................................................ 80
- `plotROC` ................................................. 82
- `predict.CauseSpecificCox` .............................. 84
- `predict.FGR` ............................................. 86
- `predict.riskRegression` ................................ 87
- `predictCox` .............................................. 88
- `predictCoxPL` .......................................... 91
- `predictRisk` ............................................ 93
- `print.ate` .............................................. 99
- `print.CauseSpecificCox` ............................... 100
- `print.FGR` .............................................. 100
- `print.influenceTest` ................................... 101
- `print.IPA` .............................................. 101
- `print.predictCox` ...................................... 102
- `print.predictCSC` ...................................... 102
- `print.riskRegression` .................................. 103
- `print.Score` ........................................... 104
- `print.subjectWeights` .................................. 104
- `reconstructData` ........................................ 105
- `riskLevelPlot` .......................................... 105
- `riskRegression` ......................................... 107
- `riskRegression.options` ................................. 110
- `rowCenter_cpp` ......................................... 111
- `rowCumProd` ............................................ 112
- `rowCumSum` ............................................. 112
- `rowMultiply_cpp` ..................................... 113
as.data.table.ate

Turn ate Object Into a data.table

Description

Turn ate object into a data.table.

Usage

```r
## S3 method for class 'ate'
as.data.table(
  x,
  keep.rownames = FALSE,
  se = TRUE,
  estimator = x$estimator,
  ...
)
```
as.data.table.influenceTest

Arguments

- **x**: object obtained with function `ate`
- **keep.rownames**: Not used.
- **se**: [logical] Should standard errors/quantile for confidence bands be displayed?
- **estimator**: [character] The type of estimator relative to which the estimates should be output.
- **...**: Not used.

Description

Turn `influenceTest` Object Into a `data.table`.

Usage

```r
## S3 method for class 'influenceTest'
as.data.table(x, keep.rownames = FALSE, se = TRUE, ...)
```

Arguments

- **x**: object obtained with function `influenceTest`
- **keep.rownames**: Not used.
- **se**: [logical] Should standard errors/quantile for confidence bands be displayed?
- **...**: Not used.

as.data.table.predictCox

Turn `predictCox` Object Into a `data.table`

Description

Turn `predictCox` object into a `data.table`.

Usage

```r
## S3 method for class 'predictCox'
as.data.table(x, keep.rownames = FALSE, se = TRUE, ...)
```
Arguments

x  object obtained with function predictCox
keep.rownames  Not used.
se  [logical] Should standard errors/quantile for confidence bands be displayed?
...  Not used.

as.data.table.predictCSC

Turn predictCSC Object Into a data.table

Description

Turn predictCSC object into a data.table.

Usage

## S3 method for class 'predictCSC'
as.data.table(x, keep.rownames = FALSE, se = TRUE, ...)

Arguments

x  object obtained with function predictCSC
keep.rownames  not used
se  should standard errors/quantile for confidence bands be displayed?
...  not used

ate

Compute the Average Treatment Effects Via

Description

Use the g-formula/IPTW/double robust estimator to estimate the average treatment effect based on Cox regression with or without competing risks.
ate

Usage

ate(
  event,
  treatment,
  censor = NULL,
  data,
  formula,
  estimator = NULL,
  strata = NULL,
  contrasts = NULL,
  times,
  cause = NA,
  landmark,
  se = TRUE,
  iid = FALSE,
  known.nuisance = FALSE,
  band = FALSE,
  B = 0,
  seed,
  handler = "foreach",
  mc.cores = 1,
  verbose = TRUE,
  ...
)

Arguments

event: Outcome model which describes how event risk depends on treatment and covariates. The object carry its own call and have a predictRisk method. See examples.

treatment: Treatment model which describes how treatment depends on covariates. The object must be a glm object (logistic regression) or the name of the treatment variable. See examples.

censor: Censoring model which describes how censoring depends on treatment and covariates. The object must be a coxph or cph object. See examples.

data: [data.frame or data.table] Data set in which to evaluate risk predictions based on the outcome model.

formula: For analyses with time-dependent covariates, the response formula. See examples.

estimator: [character] The type of estimator used to compute the average treatment effect. Can be "G-formula", "IPTW", or "AIPTW". When using estimator="G-formula", a model for the outcome should be provided (argument event). When using estimator="IPTW", a model for the treatment should be provided (argument treatment), as well as for the censoring (if any, argument censor). When using estimator="AIPTW" (double robust estimator), a model for the outcome and...
the treatment should be provided (argument event and treatment), as well as for the censoring (if any, argument censor).

strata [character] Strata variable on which to compute the average risk. Incompatible with treatment. Experimental.

contrasts [character] The levels of the treatment variable to be compared.

times [numeric vector] Time points at which to evaluate average treatment effects.

cause [integer/character] the cause of interest.

landmark for models with time-dependent covariates the landmark time(s) of evaluation. In this case, argument time may only be one value and for the prediction of risks it is assumed that the covariates do not change between landmark and landmark+time.

se [logical] If TRUE compute and add the standard errors to the output.

iid [logical] If TRUE compute and add the influence function to the output.

known.nuisance [logical] If FALSE the uncertainty related to the estimation of the nuisance parameters is ignored. This greatly simplifies computations but requires to use a double robust estimator and to assumes that all event, treatment, and censoring models are valid to obtain consistent standard errors.

band [logical] If TRUE compute and add the quantiles for the confidence bands to the output.

B [integer, >0] the number of bootstrap replications used to compute the confidence intervals. If it equals 0, then the influence function is used to compute Wald-type confidence intervals/bands.

seed [integer, >0] seed number used to generate seeds for bootstrap and to achieve reproducible results.

handler [character] Parallel handler for bootstrap. either "foreach", "mclapply", "snow" or " multicore". if "foreach" use doparallel to create a cluster.

mc.cores [integer, >0] The number of cores to use, i.e., the upper limit for the number of child processes that run simultaneously. Passed to parallel::mclapply or doparallel::registerdoparallel. The option is initialized from environment variable mc_cores if set.

c1 A parallel socket cluster used to perform cluster calculation in parallel. Output by parallel::makeCluster. The packages necessary to run the computations (e.g. riskRegression) must already be loaded on each worker.

verbose [logical] If TRUE inform about estimated run time. "minimal" requires less memory but can only estimate the standard for the difference between treatment effects (and not for the ratio).

... passed to predictRisk

Author(s)

Brice Ozenne <broz@sund.ku.dk> and Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

cconfint.ate to compute confidence intervals/bands. autoplot.ate to display the average risk.
Examples

library(survival)
library(rms)
library(prodlim)
set.seed(10)

#### Survival settings ####
#### ATE with Cox model ####
## generate data
n <- 100
dtS <- sampleData(n, outcome="survival")
dtS$time <- round(dtS$time,1)
dtS$X1 <- factor(rbinom(n, prob = c(0.3,0.4) , size = 2), labels = paste0("T",0:2))

## estimate the Cox model
fit <- cph(formula = Surv(time,event)~ X1+X2,data=dtS,y=TRUE,x=TRUE)

## compute the ATE at times 5, 6, 7, and 8 using X1 as the treatment variable
## Not run:
## only point estimate (argument se = FALSE)
ateFit1a <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = FALSE)

## standard error / confidence intervals computed using the influence function
## (argument se = TRUE and B = 0)
ateFit1b <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = TRUE, B = 0)

## same as before with in addition the confidence bands for the ATE
## (argument band = TRUE)
ateFit1c <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = TRUE, band = TRUE, B = 0)

## standard error / confidence intervals computed using 100 boostrap samples
## (argument se = TRUE and B = 100)
ateFit1d <- ate(fit, data = dtS, treatment = "X1",
               times = 5:8, se = TRUE, B = 100)

## NOTE: for real applications 100 bootstrap samples is not enough

## same but using 2 cpus for generating and analyzing the boostrap samples
## (parallel computation, argument mc.cores = 2)
ateFit1e <- ate(fit, data = dtS, treatment = "X1",
               times = 5:8, se = TRUE, B = 100, mc.cores = 2)

## End(Not run)

#### Survival settings without censoring ####
#### ATE with glm ####
## generate data
n <- 100
dtB <- sampleData(n, outcome="binary")
dtB[, X2 := as.numeric(X2)]

## estimate a logistic regression model
fit <- glm(formula = Y ~ X1+X2, data=dtB, family = "binomial")

## compute the ATE using X1 as the treatment variable
## only point estimate (argument se = FALSE)
ateFit1a <- ate(fit, data = dtB, treatment = "X1", se = FALSE)

## Not run:
## standard error / confidence intervals computed using the influence function
ateFit1b <- ate(fit, data = dtB, treatment = "X1",
              times = 5, ## just for having a nice output not used in computations
              se = TRUE, B = 0)

## standard error / confidence intervals computed using 100 bootstrap samples
ateFit1d <- ate(fit, data = dtB, treatment = "X1",
              times = 5, se = TRUE, B = 100)

## using the lava package
ateLava <- estimate(fit, function(p, data){
a <- p["(Intercept)"]; b <- p["X11"] ; c <- p["X2"] ;
R.X11 <- expit(a + b + c * data["X2"])
R.X10 <- expit(a + c * data["X2"])
list(risk0=R.X10,risk1=R.X11,riskdiff=R.X11-R.X10),
     average=TRUE)
ateLava

ateFit1b$meanRisk

## End(Not run)

#### Competing risks settings ####
#### ATE with cause specific Cox regression ####

## Not run:
## generate data
n <- 500
set.seed(10)
dt <- sampleData(n, outcome="competing.risks")
dt$time <- round(dt$time,1)
dt$X1 <- factor(rbinom(n, prob = c(0.2,0.3) , size = 2), labels = paste0("T",0:2))

## estimate cause specific Cox model
fitCR <- CSC(Hist(time,event)~ X1+X8,data=dt,cause=1)

## compute the ATE at times 10, 15, 20 using X1 as the treatment variable
ateFit2a <- ate(fitCR, data = dt, treatment = "X1", times = c(10,15,20),
               cause = 1, se = FALSE)

## standard error / confidence intervals computed using the influence function
## (argument se = TRUE and B = 0)
ate

ateFit2b <- ate(fitCR, data = dt, treatment = "X1", times = c(10,15,20),
cause = 1, se = TRUE, B = 0)

## same as before with in addition the confidence bands for the ATE
## (argument band = TRUE)
ateFit2c <- ate(fitCR, data = dt, treatment = "X1", times = c(10,15,20),
cause = 1, se = TRUE, band = TRUE, B = 0)

## standard error / confidence intervals computed using 100 bootstrap samples
## (argument se = TRUE and B = 100)
ateFit2d <- ate(fitCR, data = dt, treatment = "X1", times = c(10,15,20),
cause = 1, se = TRUE, B = 100)

## NOTE: for real applications 100 bootstrap samples is not enough
## same but using 2 cpus for generating and analyzing the bootstrap samples
## (parallel computation, argument mc.cores = 2)
ateFit2e <- ate(fitCR, data = dt, treatment = "X1", times = c(10,15,20),
cause = 1, se = TRUE, B = 100, mc.cores = 2)

## End(Not run)

#### time-dependent covariates ####
## Not run:
library(survival)
fit <- coxph(Surv(time, status) ~ celltype+karno + age + trt, veteran)
vet2 <- survSplit(Surv(time, status) ~., veteran,
cut=c(60, 120), episode ="timegroup")
fitTD <- coxph(Surv(tstart, time, status) ~ celltype+karno + age + trt,
data= vet2,x=1)
set.seed(16)
resVet <- ate(fitTD,formula=Hist(entry=tstart,time=time,event=status)~1,
data = vet2, treatment = "celltype", contrasts = NULL,
times=5,verbose=1,
landmark = c(0,30,60,90), cause = 1, B = 10, se = 1,
band = FALSE, mc.cores=1)
resVet

## End(Not run)

## Not run:
set.seed(137)
d=sampleDataTD(127)
library(survival)
d[,status:=1*(event==1)]
d[,X3:=as.factor(X3)]
## ignore competing risks
cox1TD <- coxph(Surv(start,time, status,type="counting") ~ X3+X5+X6+X8,
data=d, x = TRUE)
resTD1 <- ate(cox1TD,formula=Hist(entry=start,time=time,event=status)~1,
data = d, treatment = "X3", contrasts = NULL,
times=.5,verbose=1,
landmark = c(0,0.5,1), B = 20, se = 1,
band = FALSE, mc.cores=1)
```r
resTD1
## account for competing risks
cscTD <- CSC(Hist(time=time, event=event, entry=start) ~ X3+X5+X6+X8, data=d)
set.seed(16)
resTD <- ate(cscTD, formula=Hist(entry=start, time=time, event=event)~1, 
data = d, treatment = "X3", contrasts = NULL, 
times=.5, verbose=1, 
landmark = c(0,0.5,1), cause = 1, B = 20, se = 1, 
band = FALSE, mc.cores=1)
resTD
## End(Not run)
```

---

### autoplot.ate

#### Description

Plot average risks.

#### Usage

```r
## S3 method for class 'ate'
autoplot(
  object, 
estimator = object$estimator[1], 
  ci = FALSE, 
  band = FALSE, 
  plot = TRUE, 
  digits = 2, 
  alpha = NA, 
  ... 
)
```

#### Arguments

- **object**: Object obtained with the function `ate`.
- **estimator**: [character] The type of estimator relative to which the risks should be displayed.
- **ci**: [logical] If TRUE display the confidence intervals for the average risks.
- **band**: [logical] If TRUE display the confidence bands for the average risks.
- **plot**: [logical] Should the graphic be plotted.
- **digits**: [integer, >0] Number of decimal places.
- **alpha**: [numeric, 0-1] Transparency of the confidence bands. Argument passed to `ggplot2::geom_ribbon`.
- **...**: not used. Only for compatibility with the plot method.
autplot.predictCox

See Also

ate to compute average risks.

Examples

```r
## Not run:
library(survival)
library(rms)
library(ggplot2)
#### simulate data ####
n <- 1e2
set.seed(10)
dtS <- sampleData(n, outcome="survival")

#### Cox model ####
fit <- cph(formula = Surv(time, event) ~ X1 + X2, data = dtS, y = TRUE, x = TRUE)

#### Average treatment effect ####
seqTimes <- sort(unique(fit$y[,1]))
seqTimes5 <- seqTimes[seqTimes > 5 & seqTimes < 10]
ateFit <- ate(fit, data = dtS, treatment = "X1", contrasts = NULL,
times = seqTimes, B = 0, band = TRUE, nsim.band = 500, y = TRUE,
mc.cores = 1)

#### display ####
ggplot2::autoplot(ateFit)
outGG <- autoplot(ateFit, band = TRUE, ci = TRUE, alpha = 0.1)
dd <- as.data.frame(outGG$data[treatment == 0])
outGG$plot + facet_wrap(~treatment, labeller = label_both)
## End(Not run)
```

autplot.predictCox  
Plot Predictions From a Cox Model

Description

Plot predictions from a Cox model.

Usage

```r
## S3 method for class 'predictCox'
autplot(
  object,
  type = NULL,
  ci = FALSE,
  band = FALSE,
)```
group.by = "row",
reduce.data = FALSE,
plot = TRUE,
ylab = NULL,
digits = 2,
alpha = NA,
...)

Arguments

object Object obtained with the function predictCox.

type [character] The type of predicted value to display. Choices are: "hazard" the hazard function, "cumhazard" the cumulative hazard function, or "survival" the survival function.

ci [logical] If TRUE display the confidence intervals for the predictions.

band [logical] If TRUE display the confidence bands for the predictions.

group.by [character] The grouping factor used to color the prediction curves. Can be "row", "strata", or "covariates".

reduce.data [logical] If TRUE only the covariates that does take identical values for all observations are displayed.

plot [logical] Should the graphic be plotted.

ylab [character] Label for the y axis.

digits [integer] Number of decimal places.

alpha [numeric, 0-1] Transparency of the confidence bands. Argument passed to ggplot2::geom_ribbon.

... Not used. Only for compatibility with the plot method.

Examples

library(survival)
library(ggplot2)

#### simulate data ####
set.seed(10)
d <- sampleData(1e2, outcome = "survival")

#### Cox model ####
m.cox <- coxph(Surv(time,event)~ X1 + X2 + X3,
data = d, x = TRUE, y = TRUE)

## display baseline hazard
e.basehaz <- predictCox(m.cox)

autoplot(e.basehaz, type = "cumhazard")

## display predicted survival
Plot Predictions From a Cause-specific Cox Proportional Hazard Regression

### Description

Plot predictions from a Cause-specific Cox proportional hazard regression.

### Usage

```r
# S3 method for class 'predictCSC'
autoplot(
  object,
  ci = FALSE,
  band = FALSE,
  group.by = "row",
  reduce.data = FALSE,
  plot = TRUE,
  digits = 2,
  alpha = NA,
  ...
)
```
Arguments

- **object**: Object obtained with the function `predictCox`.
- **ci**: [logical] If TRUE display the confidence intervals for the predictions.
- **band**: [logical] If TRUE display the confidence bands for the predictions.
- **group.by**: [character] The grouping factor used to color the prediction curves. Can be "row", "strata", or "covariates".
- **reduce.data**: [logical] If TRUE only the covariates that does take identical values for all observations are displayed.
- **plot**: [logical] Should the graphic be plotted.
- **digits**: [integer] Number of decimal places.
- **alpha**: [numeric, 0-1] Transparency of the confidence bands. Argument passed to `ggplot2::geom_ribbon`.
- **...**: Not used. Only for compatibility with the plot method.

Examples

```r
library(survival)
library(rms)
library(ggplot2)
library(prodlim)

#### simulate data ####
set.seed(10)
d <- sampleData(1e2, outcome = "competing.risks")

#### CSC model ####
m.CSC <- CSC(Hist(time,event)~ X1 + X2 + X6, data = d)
pred.CSC <- predict(m.CSC, newdata = d[1:2,], time = 1:5, cause = 1)#'
autoplot(pred.CSC)

#### stratified CSC model ####
m.SCSC <- CSC(Hist(time,event)~ strata(X1) + strata(X2) + X6,
data = d)
pred.SCSC <- predict(m.SCSC, time = 1:3, newdata = d[1:4,],
cause = 1, keep.newdata = TRUE, keep.strata = TRUE)
autoplot(pred.SCSC, group.by = "strata")
```

Description

**ggplot AUC curve**
Usage

## S3 method for class 'Score'
autoplot(
  object,
  models,
  type = "score",
  lwd = 2,
  xlim,
  ylim,
  axes = TRUE,
  conf.int = FALSE,
  ...
)

Arguments

object Object obtained with Score.list
models Choice of models to plot
type Character. Either "score" to show AUC or "contrasts" to show differences between AUC.
lwd Line width
xlim Limits for x-axis
ylim Limits for y-axis
axes Logical. If TRUE draw axes.
conf.int Logical. If TRUE draw confidence shadows.
...
Not yet used

Examples

library(survival)
library(ggplot2)
d = sampleData(100, outcome = "survival")
nd = sampleData(100, outcome = "survival")
f1 = coxph(Surv(time, event) ~ X1 + X6 + X8, data = d, x = TRUE, y = TRUE)
f2 = coxph(Surv(time, event) ~ X2 + X5 + X9, data = d, x = TRUE, y = TRUE)
xx = Score(list(f1, f2), formula = Surv(time, event) ~ 1, data = nd, metrics = "auc", null.model = FALSE, times = seq(3:10))
g <- autoplot(xx)
print(g)
aucgraph <- plotAUC(xx)
plotAUC(xx, conf.int = TRUE)
plotAUC(xx, which = "contrasts")
plotAUC(xx, which = "contrasts", conf.int = TRUE)
boot2pvalue  
*Compute the p.value from the distribution under H1*

**Description**

Compute the p.value associated with the estimated statistic using a bootstrap sample of its distribution under H1.

**Usage**

```r
boot2pvalue(
  x,
  null,
  estimate = NULL,
  alternative = "two.sided",
  FUN.ci = quantileCI,
  tol = .Machine$double.eps^0.5
)
```

**Arguments**

- `x` [numeric vector] a vector of bootstrap estimates of the statistic.
- `null` numeric value of the statistic under the null hypothesis.
- `estimate` numeric the estimated statistic.
- `alternative` character a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
- `FUN.ci` function the function used to compute the confidence interval. Must take `x`, `alternative`, `conf.level` and `sign.estimate` as arguments and only return the relevant limit (either upper or lower) of the confidence interval.
- `tol` numeric the absolute convergence tolerance.

**Details**

For test statistic close to 0, this function returns 1.

For positive test statistic, this function search the quantile alpha such that:

- `quantile(x,probs = alpha)==0` when the argument alternative is set to "greater".
- `quantile(x,probs = 0.5*alpha)==0` when the argument alternative is set to "two.sided".

If the argument alternative is set to "less", it returns 1.

For negative test statistic, this function search the quantile alpha such that:
• \text{quantile}(x, \text{probs} = 1-\alpha=0) \text{ when the argument alternative is set to "less".}

• \text{quantile}(x, \text{probs} = 1-0.5\alpha=0) \text{ when the argument alternative is set to "two.sided".}

If the argument alternative is set to "greater", it returns 1.

\textbf{Examples}

\begin{verbatim}
set.seed(10)

#### no effect ####
x <- rnorm(1e3)
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "two.sided")
## expected value of 1
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "greater")
## expected value of 0.5
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "less")
## expected value of 0.5

#### positive effect ####
x <- rnorm(1e3, mean = 1)
boot2pvalue(x, null = 0, estimate = 1, alternative = "two.sided")
## expected value of 0.32 = 2*p\text{norm}(q = 0, mean = -1) = 2*mean(x<=0)
boot2pvalue(x, null = 0, estimate = 1, alternative = "greater")
## expected value of 0.16 = p\text{norm}(q = 0, mean = 1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = 1, alternative = "less")
## expected value of 0.84 = 1-p\text{norm}(q = 0, mean = 1) = mean(x>=0)

#### negative effect ####
x <- rnorm(1e3, mean = -1)
boot2pvalue(x, null = 0, estimate = -1, alternative = "two.sided")
## expected value of 0.32 = 2*(1-p\text{norm}(q = 0, mean = -1)) = 2*mean(x>=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "greater")
## expected value of 0.84 = p\text{norm}(q = 0, mean = -1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "less") # p\text{norm}(q = 0, mean = -1)
## expected value of 0.16 = 1-p\text{norm}(q = 0, mean = -1) = mean(x>=0)
\end{verbatim}

---

\textbf{Description}

Retrospective boxplots of risk quantiles conditional on outcome

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'Score'
boxplot(
  x,
  model,
\end{verbatim}
reference,  
  type = "risk",  
  timepoint,  
  overall = 1L,  
  lwd = 3,  
  xlim,  
  xlab = "",  
  main,  
  outcome.label,  
  outcome.label.offset = 0,  
  event.labels,  
  reffine = (type != "risk"),  
  add = FALSE,  
  ...
)

Arguments

  x        Score object obtained by calling function Score.
  model    Choice of risk prediction model
  reference Choice of reference risk prediction model for calculation of risk differences.
  type     Either "risk" for predicted risks or "diff" for differences between predicted risks.
  timepoint time point specifying the prediction horizon
  overall   Logical. Tag to be documented.
  lwd       line width
  xlim      x-axis limits
  xlab      x-axis label
  main      title of plot
  outcome.label Title label for column which shows the outcome status
  outcome.label.offset Vertical offset for outcome.label
  event.labels Labels for the different events (causes).
  reffine   Logical, for type="diff" only. If TRUE draw a red vertical line at 0.
  add       Logical. Tag to be documented.
  ...       not used

Examples

  # binary outcome
  library(data.table)
  library(prodlim)
  db=sampleData(40,outcome="binary")
  fitconv=glm(Y~X3+X5,data=db,family=binomial)
  fitnew=glm(Y~X1+X3+X5+X6+X7,data=db,family=binomial)
calcSeCox

Computation of standard errors for predictions

```r
x = Score(list(new=fitnew, conv=fitconv),
    formula = Y ~ 1, contrasts = list(c(2,1)),
    data = db, plots = "box", null.model = FALSE)
boxplot(x)

# survival outcome
library(survival)
ds = sampleData(40, outcome = "survival")
fit = coxph(Surv(time, event) ~ X6 + X9, data = ds, x = TRUE, y = TRUE)
## Not run:
scoreobj = Score(list("Cox" = fit),
    formula = Hist(time, event) - 1, data = ds,
    metrics = NULL, plots = "box",
    times = c(1, 5), null.model = FALSE)
boxplot(scoreobj, timepoint = 5)
boxplot(scoreobj, timepoint = 1)

## End(Not run)

# competing risks outcome
library(survival)
data(Melanoma, package = "riskRegression")
fit = CSC(Hist(time, event, cens.code = "censored") ~ invasion + age + sex, data = Melanoma)
scoreobj = Score(list("CSC" = fit),
    formula = Hist(time, event, cens.code = "censored") - 1,
    data = Melanoma, plots = "box", times = 5 * 365.25, null.model = FALSE)
par(mar = c(4, 12, 4, 4))
boxplot(scoreobj, timepoint = 5 * 365.25)

# more than 2 competing risks
m = lava::lvm(~ X1 + X2 + X3)
lava::distribution(m, "eventtime1") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "eventtime2") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "eventtime3") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "censtime") <- lava::coxWeibull.lvm(scale = 1/100)
lava::regression(m, eventtime2 ~ X3) = 1.3
m <- lava::eventTime(m,
    time = min(eventtime1 = 1, eventtime2 = 2, eventtime3 = 3, censtime = 0), "event")
set.seed(101)
dcr = as.data.table(lava::sim(m, 101))
fit = CSC(Hist(time, event) ~ X1 + X2 + X3, data = dcr)
scoreobj = Score(list("my model" = fit),
    formula = Hist(time, event) - 1,
    data = dcr, plots = "box", times = 5, null.model = FALSE)
boxplot(scoreobj)
```
calcSeCox

Description
Compute the standard error associated to the predictions from Cox regression model using a first order von Mises expansion of the functional (cumulative hazard or survival).

Usage
calcSeCox(
  object,
  times,
  nTimes,
  type,
  diag,
  Lambda0,
  object.n,
  object.time,
  object.eXb,
  object.strata,
  nStrata,
  new.n,
  new.eXb,
  new.LPdata,
  new.strata,
  new.survival,
  nVar,
  export,
  store.iid
)

Arguments

object  The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
times  Vector of times at which to return the estimated hazard/survival.
nTimes  the length of the argument times.
type  One or several strings that match (either in lower or upper case or mixtures) one or several of the strings "hazard","cumhazard","survival".
diag  [logical] when FALSE the hazard/cumulative hazard/survival for all observations at all times is computed, otherwise it is only computed for the i-th observation at the i-th time.
Lambda0  the baseline hazard estimate returned by BaseHazStrata_cpp.
object.n  the number of observations in the dataset used to estimate the object.
object.time  the time to event of the observations used to estimate the object.
object.eXb  the exponential of the linear predictor relative to the observations used to estimate the object.
object.strata  the strata index of the observations used to estimate the object.
calcSeCSC

**nStrata**
the number of strata.

**new.n**
the number of observations for which the prediction was performed.

**new.eXb**
the linear predictor evaluated for the new observations.

**new.LPdata**
the variables involved in the linear predictor for the new observations.

**new.strata**
the strata indicator for the new observations.

**new.survival**
the survival evaluated for the new observations.

**nVar**
the number of variables that form the linear predictor.

**export**
can be "iid" to return the value of the influence function for each observation. "se" to return the standard error for a given timepoint.

**store.iid**
Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal". See the details section.

**Details**
Can also return the estimated influence function for the cumulative hazard function and survival probabilities the sum over the observations of the estimated influence function.

store.iid="full" compute the influence function for each observation at each time in the argument times before computing the standard error / influence functions. store.iid="minimal" recompute for each subject specific prediction the influence function for the baseline hazard. This avoid to store all the influence functions but may lead to repeated evaluation of the influence function. This solution is therefore more efficient in memory usage but may not be in terms of computation time.

**Value**
A list optionally containing the standard error for the survival, cumulative hazard and hazard.

**Author(s)**
Brice Ozenne broz@sund.ku.dk, Thomas A. Gerds tag@biostat.ku.dk

<table>
<thead>
<tr>
<th>calcSeCSC</th>
<th><strong>Standard error of the absolute risk predicted from cause-specific Cox models</strong></th>
</tr>
</thead>
</table>

**Description**
Standard error of the absolute risk predicted from cause-specific Cox models using a first order von Mises expansion of the absolute risk functional.
Usage

calcSeCSC(
  object,
  cif,
  hazard,
  cumhazard,
  survival,
  object.time,
  object.maxtime,
  eXb,
  new.LPdata,
  new.strata,
  times,
  surv.type,
  ls.infoVar,
  new.n,
  cause,
  nCause,
  nVar,
  export,
  store.iid,
  diag
)

Arguments

  object The fitted cause specific Cox model
  cif the cumulative incidence function at each prediction time for each individual.
  hazard list containing the baseline hazard for each cause in a matrix form. Columns
    correspond to the strata.
  cumhazard list containing the cumulative baseline hazard for each cause in a matrix form. 
    Columns correspond to the strata.
  survival list containing the (all cause) survival in a matrix form. Columns correspond to 
    event times.
  object.time a vector containing all the events regardless to the cause.
  object.maxtime a matrix containing the latest event in the strata of the observation for each cause.
  eXb a matrix containing the exponential of the linear predictor evaluated for the new 
    observations (rows) for each cause (columns)
  new.LPdata a list of design matrices for the new observations for each cause.
  new.strata a matrix containing the strata indicator for each observation and each cause.
  times the time points at which to evaluate the predictions.
  surv.type see the surv.type argument of CSC.
  ls.infoVar A list containing the output of coxVariableName for each Cox model.
  new.n the number of new observations.
cause the cause of interest.
nCause the number of causes.
nVar the number of variables that form the linear predictor in each Cox model
export can be "iid" to return the value of the influence function for each observation 
"se" to return the standard error for a given timepoint
store.iid the method used to compute the influence function and the standard error. Can 
be "full" or "minimal". See the details section.
diag [logical] when FALSE the absolute risk/survival for all observations at all times 
is computed, otherwise it is only computed for the i-th observation at the i-th 
time.

Details
Can also return the empirical influence function of the functionals cumulative hazard or survival or 
the sum over the observations of the empirical influence function.
store.iid="full" compute the influence function for each observation at each time in the ar-
argument times before computing the standard error / influence functions. store.iid="minimal" 
recompute for each subject specific prediction the influence function for the baseline hazard. This 
avoid to store all the influence functions but may lead to repeated evaluation of the influence func-
tion. This solution is therefore efficient more efficient in memory usage but may not be in term of 
computation time.

Cforest

S3-wrapper function for cforest from the party package

Description
S3-wrapper function for cforest from the party package

Usage
Cforest(formula, data, ...)

Arguments

formula Passed on as is. See cforest of the party package
data Passed on as is. See cforest of the party package
... Passed on as they are. See cforest of the party package

Details
See cforest of the party package.

Value
list with two elements: cforest and call
References


---

coef.CauseSpecificCox  Extract coefficients from a Cause-Specific Cox regression model

Description

Extract coefficients from a Cause-Specific Cox regression model

Usage

```r
## S3 method for class 'CauseSpecificCox'
coef(object, ...)
```

Arguments

- `object`: Object obtained with CSC
- `...`: not used

---

coef.riskRegression  Extract coefficients from riskRegression model

Description

Extract coefficients from riskRegression model

Usage

```r
## S3 method for class 'riskRegression'
coef(object, digits = 3, eps = 10^-4, ...)
```

Arguments

- `object`: Object obtained with ARR or LRR or riskRegression
- `digits`: Number of digits
- `eps`: P-values below this number are shown as <eps
- `...`: not used
**colCenter_cpp**

**Apply - by column**

**Description**

Fast computation of sweep(X, MARGIN = 1, FUN = "-", STATS = center)

**Usage**

```r
colCenter_cpp(X, center)
```

**Arguments**

- `X` A matrix.
- `center` a numeric vector of length equal to the number of rows of `x`

**Value**

A matrix of same size as X.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```r
x <- matrix(1,6,5)
sweep(x, MARGIN = 1, FUN = "-", STATS = 1:6)
colCenter_cpp(x, 1:6)
```

**colCumProd**

**Apply cumprod in each column**

**Description**

Fast computation of apply(x,2,cumprod)

**Usage**

```r
colCumProd(x)
```

**Arguments**

- `x` A matrix.
Value

A matrix of same size as x.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```r
x <- matrix(1:8,ncol=2)
colCumProd(x)
```

---

Description

Fast computation of `apply(x,2,cumsum)`

Usage

```r
colCumSum(x)
```

Arguments

- `x` A matrix.

Value

A matrix of same size as x.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```r
x <- matrix(1:8,ncol=2)
colCumSum(x)
```
**colMultiply_cpp**  
*Apply * by column*

**Description**  
Fast computation of sweep(X, MARGIN = 1, FUN = "*", STATS = scale)

**Usage**  
```r  
colMultiply_cpp(X, scale)  
```  

**Arguments**  
- **X**  
  A matrix.  
- **scale**  
  a numeric vector of length equal to the number of rows of x

**Value**  
A matrix of same size as X.

**Author(s)**  
Brice Ozenne <broz@sund.ku.dk>

**Examples**  
```r  
x <- matrix(1,6,5)  
sweep(x, MARGIN = 1, FUN = "*", STATS = 1:6)  
colMultiply_cpp(x, 1:6 )  
```

---

**colScale_cpp**  
*Apply / by column*

**Description**  
Fast computation of sweep(X, MARGIN = 1, FUN = "/", STATS = scale)

**Usage**  
```r  
colScale_cpp(X, scale)  
```  

**Arguments**  
- **X**  
  A matrix.  
- **scale**  
  a numeric vector of length equal to the number of rows of x
colSumsCrossprod

Value

A matrix of same size as X.

Author(s)

Brice Ozenne <broz@sund.ku.dk>

Examples

```r
x <- matrix(1,6,5)
sweep(x, MARGIN = 1, FUN = "/", STATS = 1:6)
colScale_cpp(x, 1:6)
```

---

Description

Apply crossprod and colSums

Usage

```r
colSumsCrossprod(X, Y, transposeY)
```

Arguments

- **X**: A matrix with dimensions k*n. Hence the result of `colSums(X)` has length n.
- **Y**: A matrix with dimensions n*m. Can be a matrix with dimension m*n but then `transposeY` should be TRUE.
- **transposeY**: Logical. If TRUE transpose Y before matrix multiplication.

Value

A vector of length m.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```r
x <- matrix(1:8,ncol=2)
y <- matrix(1:16,ncol=8)
colSumsCrossprod(x,y,0)

x <- matrix(1:8,ncol=2)
y <- matrix(1:16,ncol=2)
colSumsCrossprod(x,y,1)
```
confBandCox

Description

Compute quantiles of a gaussian process

Usage

cconfBandCox(iid, se, n.sim, conf.level)

Arguments

- **iid**: The iid decomposition of the estimator over time.
- **se**: The variance of the estimate over time.
- **n.sim**: The number of simulations used to compute the quantiles.
- **conf.level**: Level of confidence.

confint.ate

Description

Confidence intervals and confidence Bands for the predicted absolute risk (cumulative incidence function).

Usage

```r
# S3 method for class 'ate'
confint(
  object,
  parm = NULL,
  level = 0.95,
  nsim.band = 10000,
  meanRisk.transform = "none",
  diffRisk.transform = "none",
  ratioRisk.transform = "none",
  seed = NA,
  bootci.method = "perc",
  ...
)
```
Arguments

object
parm
level
nsim.band
meanRisk.transform
diffRisk.transform
ratioRisk.transform
seed
bootci.method
...

Details

Confidence bands and confidence intervals computed via the influence function are automatically restricted to the interval [0;1].

Confidence intervals obtained via bootstrap are computed using the boot.ci function of the boot package. p-value are obtained using test inversion method (finding the smallest confidence level such that the interval contain the null hypothesis).

Author(s)

Brice Ozenne

Examples

library(survival)
library(data.table)

## ## generate data ####
set.seed(10)
d <- sampleData(70,outcome="survival")
d[, X1 := paste0("T",rbinom(.N, size = 2, prob = c(0.51)))]
## table(d$X1)
### stratified Cox model ###
fit <- coxph(Surv(time,event)~X1 + strata(X2) + X6,
  data=d, ties="breslow", x = TRUE, y = TRUE)

### average treatment effect ###
fit.ate <- ate(fit, treatment = "X1", times = 1:3, data = d,
  se = TRUE, iid = TRUE, band = TRUE)
print(fit.ate, type = "meanRisk")
dt.ate <- as.data.table(fit.ate)

## manual calculation of se
dd <- copy(d)
dd$X1 <- rep(factor("T0", levels = paste0("T",0:2)), NROW(dd))
out <- predictCox(fit, newdata = dd, se = TRUE, times = 1:3, average.iid = TRUE)
term1 <- -out$survival.average.iid
term2 <- sweep(1-out$survival, MARGIN = 2, FUN = "-", STATS = colMeans(1-out$survival))
sqrt(colSums((term1 + term2/NROW(d))^2))
## fit.ate$meanRisk[treatment=="T0",meanRisk.se]

## check confidence intervals (no transformation)
dt.ate[,.(lower = pmax(0,value + qnorm(0.025) * se),
  lower2 = lower,
  upper = value + qnorm(0.975) * se,
  upper2 = upper)]

## add confidence intervals computed on the log-log scale
## and backtransformed
outCI <- confint(fit.ate,
  meanRisk.transform = "loglog", diffRisk.transform = "atanh",
  ratioRisk.transform = "log")
print(outCI, type = "meanRisk")
dt.ate[type == "ate", newse := se/(value*log(value))]
dt.ate[type == "ate", (lower = exp(-exp(log(-log(value)) - 1.96 * newse)),
  upper = exp(-exp(log(-log(value)) + 1.96 * newse)))]
### Usage

```r
## S3 method for class 'influenceTest'
confint(
  object,
  parm = NULL,
  level = 0.95,
  nsim.band = 10000,
  transform = "none",
  seed = NA,
  ...
)
```

### Arguments

- **object**: A `influenceTest` object, i.e. output of the `influenceTest` function.
- **parm**: not used. For compatibility with the generic method.
- **level**: [numeric, 0-1] Level of confidence.
- **nsim.band**: [integer, >0] the number of simulations used to compute the quantiles for the confidence bands.
- **transform**: [character] the transformation used to improve coverage of the confidence intervals. Can be "none" or "atanh".
- **seed**: [integer, >0] seed number set before performing simulations for the confidence bands. If not given or NA no seed is set.
- **...**: not used.

### Details

Except for the cumulative hazard, the confidence bands and confidence intervals are automatically restricted to the interval [-1;1].

### Author(s)

Brice Ozenne

---

**confint.predictCox**

Confidence Intervals and Confidence Bands for the predicted Survival/Cumulative Hazard

**Description**

Confidence intervals and confidence Bands for the predicted survival/cumulative Hazard.
Usage

```r
## S3 method for class 'predictCox'
confint(
  object,
  parm = NULL,
  level = 0.95,
  nsim.band = 10000,
  cumhazard.transform = "log",
  survival.transform = "loglog",
  seed = NA,
  ...
)
```

Arguments

- `object`: A `predictCox` object, i.e. output of the `predictCox` function.
- `parm`: [character] the type of predicted value for which the confidence intervals should be output. Can be "survival" or "cumhazard".
- `level`: [numeric, 0-1] Level of confidence.
- `nsim.band`: [integer, >0] the number of simulations used to compute the quantiles for the confidence bands.
- `cumhazard.transform`: [character] the transformation used to improve coverage of the confidence intervals for the cumulative hazard in small samples. Can be "none", "log".
- `survival.transform`: [character] the transformation used to improve coverage of the confidence intervals for the survival in small samples. Can be "none", "log", "loglog", "cloglog".
- `seed`: [integer, >0] seed number set before performing simulations for the confidence bands. If not given or NA no seed is set.
- `...`: not used.

Details

The confidence bands and confidence intervals are automatically restricted to the interval of definition of the statistic, i.e. a confidence interval for the survival of [0.5;1.2] will become [0.5;1].

Author(s)

Brice Ozenne

Examples

```r
library(survival)

#### generate data ####
set.seed(10)
```
### S3 method for class 'predictCSC'

```r
c conflint <- predictCox(fit, newdata = d[1:3], times = c(3, 8), type = "survival", 
    se = TRUE, iid = TRUE, band = TRUE)
c conflint
```

## check standard error

```r
sqrt(rowSums(conflint$survival.iid[,1]^2)) ## se for individual 1
```

## check confidence interval

```r
newse <- conflint$survival.se/(-conflint$survival*log(conflint$survival))
cbind(lower = as.double(exp(-exp(log(-log(conflint$survival)) + 1.96 * newse))), 
    upper = as.double(exp(-exp(log(-log(conflint$survival)) - 1.96 * newse)))
)
```

### compute confidence intervals without transformation

```r
c conflint(conflint$survival, transform = "none")
cbind(lower = as.double(conflint$survival - 1.96 * conflint$survival.se), 
    upper = as.double(conflint$survival + 1.96 * conflint$survival.se)
)
```

---

**confint.predictCSC**  
**Confidence Intervals and Confidence Bands for the Predicted Absolute Risk (Cumulative Incidence Function)**

### Description

Confidence intervals and confidence bands for the predicted absolute risk (cumulative incidence function).

### Usage

```r
## S3 method for class 'predictCSC'
conflint(
  object, 
  parm = NULL, 
  level = 0.95, 
  nsim.band = 10000, 
  absRisk.transform = "loglog", 
  seed = NA, 
  ... 
)
```
Arguments

- **object**: A `predictCSC` object, i.e. output of the `predictCSC` function.
- **parm**: not used. For compatibility with the generic method.
- **level**: [numeric, 0-1] Level of confidence.
- **nsim.band**: [integer, >0] the number of simulations used to compute the quantiles for the confidence bands.
- **absRisk.transform**: [character] the transformation used to improve coverage of the confidence intervals for the predicted absolute risk in small samples. Can be "none", "log", "loglog", "cloglog".
- **seed**: [integer, >0] seed number set before performing simulations for the confidence bands. If not given or NA no seed is set.
- **...**: not used.

Details

The confidence bands and confidence intervals are automatically restricted to the interval [0;1].

Author(s)

Brice Ozenne

Examples

```r
library(survival)
library(prodlim)

#### generate data ####
set.seed(10)
d <- sampleData(100)

#### estimate a stratified CSC model ####
fit <- CSC(Hist(time,event)~ X1 + strata(X2) + X6, data=d)

#### compute individual specific risks
fit.pred <- predict(fit, newdata=d[1:3], times=c(3,8), cause = 1, se = TRUE, iid = TRUE, band = TRUE)
fit.pred

#### check confidence intervals
newse <- fit.pred$absRisk.se/(fit.pred$absRisk*log(fit.pred$absRisk))
cbind(lower = as.double(exp(-exp(log(-log(fit.pred$absRisk)) + 1.96 * newse))),
       upper = as.double(exp(-exp(log(-log(fit.pred$absRisk)) - 1.96 * newse)))
)

#### compute confidence intervals without transformation
confint(fit.pred, absRisk.transform = "none")
cbind(lower = as.double(fit.pred$absRisk - 1.96 * fit.pred$absRisk.se),
       upper = as.double(fit.pred$absRisk + 1.96 * fit.pred$absRisk.se)
)
```
**coxBaseEstimator**  
*Extract the type of estimator for the baseline hazard*

**Description**
Extract the type of estimator for the baseline hazard

**Usage**
```r
coxBaseEstimator(object)
## S3 method for class 'coxph'
coxBaseEstimator(object)
## S3 method for class 'phreg'
coxBaseEstimator(object)
```

**Arguments**
- `object` The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).

**Author(s)**
Brice Ozenne broz@sund.ku.dk

---

**coxCenter**  
*Extract the mean value of the covariates*

**Description**
Extract the mean value of the covariates

**Usage**
```r
coxCenter(object)
## S3 method for class 'cph'
coxCenter(object)
## S3 method for class 'coxph'
coxCenter(object)
## S3 method for class 'phreg'
coxCenter(object)
```
 Arguments

 object  The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

 Author(s)

 Brice Ozenne broz@sund.ku.dk

---

### coxFormula

#### Description

Extract the formula from a Cox model

#### Usage

```r
coxFormula(object)
```

## S3 method for class 'cph'
coxFormula(object)

## S3 method for class 'coxph'
coxFormula(object)

## S3 method for class 'phreg'
coxFormula(object)

## S3 method for class 'glm'
coxFormula(object)

 Arguments

 object  The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

 Author(s)

 Brice Ozenne broz@sund.ku.dk
**Description**
Compute the linear predictor of a Cox model

**Usage**
coxLP(object, data, center)

```r
coxLP(object, data, center)
# S3 method for class 'cph'
coxLP(object, data, center)
# S3 method for class 'coxph'
coxLP(object, data, center)
# S3 method for class 'phreg'
coxLP(object, data, center)
```

**Arguments**
- `object` The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).
- `data` a `data.frame` or a `data.table`
- `center` should the linear predictor be computed after centering the covariates

**Details**
In case of empty linear predictor returns a vector of 0 with the same length as the number of rows of the dataset

**Author(s)**
Brice Ozenne broz@sund.ku.dk

---

**Description**
Extract the design matrix used to train a Cox model. Should contain the time of event, the type of event, the variable for the linear predictor, the strata variables and the date of entry (in case of delayed entry).
Usage

```r
coxModelFrame(object, center)
## S3 method for class 'coxph'
coxModelFrame(object, center = FALSE)
## S3 method for class 'cph'
coxModelFrame(object, center = FALSE)
## S3 method for class 'phreg'
coxModelFrame(object, center = FALSE)
```

Arguments

- `object` The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).
- `center` [logical] Should the variables of the linear predictor be added?

Author(s)

Brice Ozenne broz@sund.ku.dk

---

**coxN**

*Extract the number of observations from a Cox model*

Description

Extract the number of observations from a Cox model

Usage

```r
coxN(object)
## S3 method for class 'cph'
coxN(object)
## S3 method for class 'coxph'
coxN(object)
## S3 method for class 'phreg'
coxN(object)
## S3 method for class 'CauseSpecificCox'
coxN(object)
## S3 method for class 'glm'
coxN(object)
```
Arguments

object       The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

Author(s)

Brice Ozenne broz@sund.ku.dk

description

Return the special character(s) of the Cox model, e.g. used to indicate the strata variables.

Usage

coxSpecial(object)

## S3 method for class 'coxph'
coxSpecial(object)

## S3 method for class 'cph'
coxSpecial(object)

## S3 method for class 'phreg'
coxSpecial(object)

Arguments

object       The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

Details

Must return a list with at least one element strata indicating the character in the formula marking the variable(s) defining the strata.

Author(s)

Brice Ozenne broz@sund.ku.dk
coxStrata

Define the strata for a new dataset

Description

Define the strata in a dataset to match those of a stratified Cox model

Usage

coxStrata(object, data, sterms, strata.vars, strata.levels)

## S3 method for class 'cph'
coxStrata(object, data, sterms, strata.vars, strata.levels)

## S3 method for class 'coxph'
coxStrata(object, data, sterms, strata.vars, strata.levels)

## S3 method for class 'phreg'
coxStrata(object, data, sterms, strata.vars, strata.levels)

Arguments

object The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
data a data.frame or a data.table
sterms terms in the formula corresponding to the strata variables
strata.vars the name of the variables used to define the strata
strata.levels a named list containing for each variable used to form the strata all its possible levels
levels the strata levels that have been used to fit the Cox model

Details

if no strata variables returns a vector of "1" (factor).

Author(s)

Brice Ozenne broz@sund.ku.dk
### coxStrataLevel

*Returns the name of the strata in Cox model*

**Description**
Return the name of the strata in Cox model

**Usage**
```r
ccoxStrataLevel(object)
```

#### Arguments
- `object` The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).

#### Author(s)
Brice Ozenne broz@sund.ku.dk

### coxVarCov

*Extract the variance covariance matrix of the beta from a Cox model*

**Description**
Extract the variance covariance matrix of the beta from a Cox model

**Usage**
```r
coxVarCov(object)
```

#### Arguments
- `object` The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).
**Arguments**

- **object**: The fitted Cox regression model object either obtained with `coxph` (survival package), `cph` (rms package), or `phreg` (mets package).

**Details**

Should return **NULL** if the Cox model has no covariate. The rows and columns of the variance covariance matrix must be named with the names used in the design matrix.

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

**Description**

Extract the name of the variables belonging to the linear predictor or used to form the strata

**Usage**

```r
coxVariableName(object, model.frame)
```

**Arguments**

- **object**: The fitted Cox regression model object either obtained with `coxph` (survival package) or `cph` (rms package).
- **model.frame**: [data.frame] dataset containing all the relevant variables (entry, time to event, type of event, variables in the linear predictor, strata). Output from `coxModelFrame`.

**Author(s)**

Brice Ozenne broz@sund.ku.dk
Description

Interface for fitting cause-specific Cox proportional hazard regression models in competing risk.

Usage

CSC(formula, data, cause, surv.type = "hazard", fitter = "coxph", ...)

Arguments

- **formula**: Either a single `Hist` formula or a list of formulas. If it is a list it must contain as many `Hist` formulas as there are causes when `surv.type="hazard"` and exactly two formulas when `surv.type="survival"`. If it is a list the first formula is used for the cause of interest specific Cox regression and the other formula(s) either for the other cause specific Cox regression(s) or for the Cox regression of the combined event where each cause counts as event. Note that when only one formula is given the covariates enter in exactly the same way into all Cox regression analyses.
- **data**: A data in which to fit the models.
- **cause**: The cause of interest. Defaults to the first cause (see Details).
- **surv.type**: Either "hazard" (the default) or "survival". If "hazard" fit cause-specific Cox regression models for all causes. If "survival" fit one cause-specific Cox regression model for the cause of interest and also a Cox regression model for event-free survival.
- **fitter**: Routine to fit the Cox regression models. If `coxph` use `survival::coxph` else use `rms::cph`.
- **...**: Arguments given to `coxph`.

Details

The causes and their order are determined by `prodlim::getStates()` applied to the `Hist` object.

Value

- **models**: a list with the fitted (cause-specific) Cox regression objects
- **response**: the event history response
- **eventTimes**: the sorted (unique) event times
- **surv.type**: the value of `surv.type`
- **theCause**: the cause of interest. see `cause`
- **causes**: the other causes
CSC

Author(s)
Thomas A. Gerds <tag@biostat.ku.dk> and Ulla B. Mogensen

References

See Also
coxph

Examples
library(prodlim)
library(survival)
data(Melanoma)
## fit two cause-specific Cox models
## different formula for the two causes
fit1 <- CSC(list(Hist(time,status)~sex+age,Hist(time,status)~invasion+epicel+log(thick)),
            data=Melanoma)
print(fit1)
## Not run:
library(Publish)
publish(fit1)
## End(Not run)

## model hazard of all cause mortality instead of hazard of type 2
fit1a <- CSC(list(Hist(time,status)~sex+age,Hist(time,status)~invasion+epicel+log(thick)),
             data=Melanoma,
             surv.type="surv")

## the predicted probabilities are similar
plot(predictRisk(fit1,times=500,cause=1,newdata=Melanoma),
      predictRisk(fit1a,times=500,cause=1,newdata=Melanoma))

## special case where cause 2 has no covariates
fit1b <- CSC(list(Hist(time,status)~sex+age,Hist(time,status)~1),
            data=Melanoma)
print(fit1b)
predict(fit1b,cause=1,times=100,newdata=Melanoma)

## same formula for both causes
fit2 <- CSC(Hist(time,status)~invasion+epicel+age,
## combine a cause-specific Cox regression model for cause 2
## and a Cox regression model for the event-free survival:
## different formula for cause 2 and event-free survival
fit3 <- CSC(list(Hist(time,status)~sex+invasion+epicel+age,
                 Hist(time,status)~invasion+epicel+age),
           surv.type="surv",
           data=Melanoma)
print(fit3)

## same formula for both causes
fit4 <- CSC(Hist(time,status)~invasion+epicel+age,
          data=Melanoma,
          surv.type="surv")
print(fit4)

## strata
fit5 <- CSC(Hist(time,status)~invasion+epicel+age+strata(sex),
           data=Melanoma,
           surv.type="surv")
print(fit5)

## sanity checks
cox1 <- coxph(Surv(time,status==1)~invasion+epicel+age+strata(sex),data=Melanoma)
cox2 <- coxph(Surv(time,status!=0)~invasion+epicel+age+strata(sex),data=Melanoma)
all.equal(coef(cox1),coef(fit5$models[[1]]))
all.equal(coef(cox2),coef(fit5$models[[2]]))

## predictions
## surv.type = "hazard": predictions for both causes can be extracted
## from the same fit
fit2 <- CSC(Hist(time,status)~invasion+epicel+age, data=Melanoma)
predict(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predictRisk(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predictRisk(fit2,cause=2,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predict(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predict(fit2,cause=2,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))

## surv.type = "surv" we need to change the cause of interest
library(survival)
fit5.2 <- CSC(Hist(time,status)~invasion+epicel+age+strata(sex),
               data=Melanoma,
               surv.type="surv",cause=2)
## now this does not work
try(predictRisk(fit5.2,cause=1,newdata=Melanoma,times=4))

## but this does
predictRisk(fit5.2,cause=2,newdata=Melanoma,times=100)
predict(fit5.2,cause=2,newdata=Melanoma,times=100)
predict(fit5.2, cause=2, newdata=Melanoma[,], times=100)

---

**Ctree**

*S3-Wrapper for ctree.*

### Description

The call is added to an ctree object

### Usage

Ctree(...)  

### Arguments

... passed to ctree  

### Value

list with two elements: ctree and call  

### Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

### See Also

Cforest  

### Examples

```r  
library(prodlim)  
library(party)  
library(survival)  
set.seed(50)  
d <- SimSurv(50)  
nd <- data.frame(X1=c(0,1,0), X2=c(-1,0,1))  
f <- Ctree(Surv(time,status)~X1+X2, data=d)  
predictRisk(f, newdata=nd, times=c(3,8))  
```
discreteRoot  

Dichotomic search for monotone function

Description
Find the root of a monotone function on a discrete grid of value using dichotomic search

Usage
```r
discreteRoot(  
  fn,  
  grid,  
  increasing = TRUE,  
  check = TRUE,  
  tol = .Machine$double.eps^0.5  
)
```

Arguments
- `fn` [function] objective function to minimize in absolute value.
- `increasing` [logical] is the function fn increasing?
- `check` [logical] should the program check that fn takes a different sign for the first vs. the last value of the grid?
- `tol` [numeric] the absolute convergence tolerance.

FGR  

Formula wrapper for crr from cmprsk

Description
Formula interface for Fine-Gray regression competing risk models.

Usage
```r
FGR(formula, data, cause = 1, y = TRUE, ...)
```

Arguments
- `formula` A formula whose left hand side is a Hist object – see Hist. The right hand side specifies (a linear combination of) the covariates. See examples below.
- `data` A data.frame in which all the variables of formula can be interpreted.
- `cause` The failure type of interest. Defaults to 1.
- `y` logical value: if TRUE, the response vector is returned in component response.
- `...`
Details

Formula interface for the function crr from the cmprsk package.

The function crr allows to multiply some covariates by time before they enter the linear predictor. This can be achieved with the formula interface, however, the code becomes a little cumbersome. See the examples.

Value

See crr.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

References

Gerds, TA and Scheike, T and Andersen, PK (2011) Absolute risk regression for competing risks: interpretation, link functions and prediction Research report 11/7. Department of Biostatistics, University of Copenhagen

See Also

riskRegression

Examples

```r
library(prodlim)
library(survival)
library(cmprsk)
library(lava)
d <- prodlim::SimCompRisk(100)
f1 <- FGR(Hist(time,cause)~X1+X2,data=d)
print(f1)

## crr allows that some covariates are multiplied by
## a function of time (see argument tf of crr)
## by FGR uses the identity matrix
f2 <- FGR(Hist(time,cause)-cov2(X1)+X2,data=d)
print(f2)

## same thing, but more explicit:
f3 <- FGR(Hist(time,cause)-cov2(X1)+cov1(X2),data=d)
print(f3)

## both variables can enter cov2:
f4 <- FGR(Hist(time,cause)-cov2(X1)+cov2(X2),data=d)
print(f4)

## change the function of time
qFun <- function(x){x^2}
```
getSplitMethod <- function(x){x}
sqFun <- function(x){x^0.5}

## multiply X1 by time^2 and X2 by time:
f5 <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2),data=d)
print(f5)
print(f5$crrFit)

## same results as crr
with(d,crr(ftime=time,
    fstatus=cause,
    cov2=d[,c("X1","X2")],
    tf=function(time){cbind(qFun(time),time)}))

## still same result, but more explicit
f5a <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2,tf=noFun),data=d)
f5a$crrFit

## multiply X1 by time^2 and X2 by sqrt(time)

f5b <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2,tf=sqFun),data=d,cause=1)

## additional arguments for crr
f6c <- FGR(Hist(time,cause)~X1+X2, data=d, cause=1, gtol=1e-5)
f6c
f6a <- FGR(Hist(time,cause)~X1+X2, data=d, cause=1, gtol=0.1)
f6a

---

**getSplitMethod**  
*Input for data splitting algorithms*

**Description**

Parse hyperparameters for data splitting algorithm

**Usage**

getSplitMethod(split.method, B, N, M, seed)

**Arguments**

- **split.method** A character string specifying the algorithm for data splitting:
  - "loob" leave one out bootstrap
  - "bootcv" bootstrap cross validation
  - "cv5" 5-fold cross validation
  - "loocv" leave one out cross validation aka N-1 fold cross validation
  - "632plus" Efron’s 632+ bootstrap

- **B** Number of repetitions of bootstrap or k-fold cross-validation

- **N** Sample size

- **M** Subsample size. Default is N (no subsampling).

- **seed** Integer passed to set.seed. If not given or NA no seed is set.
Value

A list with the following elements:

- split.methodName: the print name of the algorithm
- split.method: the internal name of the algorithm
- index: the index for data splitting. For bootstrap splitting this is a matrix with B columns and M rows identifying the in-bag subjects. For k-fold cross-validation this is a matrix with B columns identifying the membership to the k groups.
- k: the k of k-fold cross-validation
- N: the sample size
- M: the subsample size

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

Score

Examples

# 3-fold crossvalidation
getSplitMethod("cv3",B=4,N=37)

# bootstrap with replacement
getSplitMethod("loob",B=4,N=37)

# bootstrap without replacement
getSplitMethod("loob",B=4,N=37,M=20)
## S3 method for class 'coxph'
iidCox(
  object,
  newdata = NULL,
  baseline.iid = TRUE,
  tau.hazard = NULL,
  tau.max = NULL,
  store.iid = "full",
  keep.times = TRUE,
  return.object = TRUE
)

## S3 method for class 'cph'
iidCox(
  object,
  newdata = NULL,
  baseline.iid = TRUE,
  tau.hazard = NULL,
  tau.max = NULL,
  store.iid = "full",
  keep.times = TRUE,
  return.object = TRUE
)

## S3 method for class 'phreg'
iidCox(
  object,
  newdata = NULL,
  baseline.iid = TRUE,
  tau.hazard = NULL,
  tau.max = NULL,
  store.iid = "full",
  keep.times = TRUE,
  return.object = TRUE
)

## S3 method for class 'CauseSpecificCox'
iidCox(
  object,
  newdata = NULL,
  baseline.iid = TRUE,
  tau.hazard = NULL,
  tau.max = NULL,
store.iid = "full",
keep.times = TRUE,
return.object = TRUE
)

Arguments

object            object The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
newdata          [data.frame] Optional new data at which to do iid decomposition
baseline.iid     [logical] Should the influence function for the baseline hazard be computed.
tau.hazard       [numeric vector] the vector of times at which the i.i.d decomposition of the baseline hazard will be computed
tau.max          [numeric] latest time at which the i.i.d decomposition of the baseline hazard will be computed. Alternative to tau.hazard.
store.iid        [character] the method used to compute the influence function and the standard error. Can be "full", "approx" or "minimal". See the details section.
keep.times       [logical] If TRUE add the evaluation times to the output.
return.object    [logical] If TRUE return the object where the iid decomposition has been added. Otherwise return a list (see the return section)

Details

This function implements the first three formula (no number,10,11) of the subsection "Empirical estimates" in (Ozenne et al., 2017).

If there is no event in a strata, the influence function for the baseline hazard is set to 0.

store.iid equal to "full" exports the influence function for the coefficients and the baseline hazard at each event time. store.iid equal to "approx" does the same except that the terms that do not contribute to the variance are not ignored (i.e. set to 0) store.iid equal to "minimal" exports the influence function for the coefficients. For the baseline hazard it only computes the quantities necessary to compute the influence function in order to save memory.

Value

For Cox models, the iid slot is a list containing:

- IFbetaInfluence function for the regression coefficient.
- IFhazardTime differential of the influence function of the hazard.
- IFcumhazardInfluence function of the cumulative hazard.
- calcIFhazardElements used to compute the influence function at a given time.
- timeTimes at which the influence function has been evaluated.
- etime1.minTime of first event (i.e. jump) in each strata.
- etime.maxLast observation time (i.e. jump or censoring) in each strata.
- indexObsIndex of the observation in the original dataset.

For Cause-Specific Cox models, a list containing the iid decomposition relative to each cause is returned.
References


Examples

```r
library(survival)
library(data.table)
library(prodlim)
set.seed(10)
d <- sampleData(100, outcome = "survival")[,.(eventtime,event,X1,X6)]
setkey(d, eventtime)

m.cox <- coxph(Surv(eventtime, event) ~ X1+X6, data = d, y = TRUE, x = TRUE)
system.time(IF.cox <- iidCox(m.cox))
system.time(IF.cox_approx <- iidCox(m.cox, store.iid = "approx"))

IF.cox.all <- iidCox(m.cox, tau.hazard = sort(unique(c(7,d$eventtime))))
IF.cox.beta <- iidCox(m.cox, baseline.iid = FALSE)
```

---

influenceTest

Influence test [Experimental!!]

Description

Compare two estimates using their influence function

Usage

```r
influenceTest(object, ...)

## S3 method for class 'list'
influenceTest(
  object,
  newdata,
  times,
  type,
  cause,
  keep.newdata = TRUE,
  keep.strata = FALSE,
  ...)

## Default S3 method:
influenceTest(object, object2, band = TRUE, ...)
```
influenceTest

Arguments

object
either a list of models or an object of class predictCox or predictCSC.

...additional arguments to be passed to lower level functions.

newdata
[data.frame or data.table] Contain the values of the predictor variables defining subject specific predictions.

times
[numeric vector] Time points at which to return the estimated absolute risk.

type
[character] the type of predicted value.

cause
[integer/character] Identifies the cause of interest among the competing events.

keep.newdata
[logical] If TRUE add the value of the covariates used to make the prediction in the output.

keep.strata
[logical] If TRUE add the value of the strata used to make the prediction in the output.

object2
same as predict1 but for another model.

band
[logical] If TRUE add the influence function to the output such that confint will be able to compute the confidence bands.

Examples

library(lava)
library(survival)
library(prodlim)
library(data.table)
n <- 100

#### Under H1
set.seed(1)
newdata <- data.frame(X1=0:1)

## simulate non proportional hazard using lava
m <- lvm()
regression(m) <- y ~ 1
regression(m) <- s ~ exp(-2*X1)
distribution(m,~X1) <- binomial.lvm()
distribution(m,~cens) <- coxWeibull.lvm(scale=1)
distribution(m,~y) <- coxWeibull.lvm(scale=1,shape=-s)
eventTime(m) <- eventtime ~ min(y=1,cens=0)
d <- as.data.table(sim(m,n))
setkey(d, eventtime)

## fit cox models
m.cox <- coxph(Surv(eventtime, status) ~ X1, data = d, y = TRUE, x = TRUE)
mStrata.cox <- coxph(Surv(eventtime, status) ~ strata(X1), data = d, y = TRUE, x = TRUE)

## compare models
# one time point
outIF <- influenceTest(list(m.cox, mStrata.cox),
  type = "survival", newdata = newdata, times = 0.5)
confint(outIF)

# several timepoints
outIF <- influenceTest(list(m.cox, mStrata.cox),
  type = "survival", newdata = newdata, times = c(0.5,1,1.5))
confint(outIF)

#### Under H0 (Cox) ####
set.seed(1)
## simulate proportional hazard using lava
m <- lvm()
regression(m,~X1) <- y ~ 1
distribution(m,~cens) <- coxWeibull.lvm()
distribution(m,~y) <- coxWeibull.lvm()
eventTime(m) <- eventtime ~ min(y=1,cens=0)
d <- as.data.table(sim(m,n))
setkey(d, eventtime)

## fit cox models
Utime <- sort(unique(d$eventtime))
m.cox <- coxph(Surv(eventtime, status) ~ X1,
  data = d, y = TRUE, x = TRUE)
mStrata.cox <- coxph(Surv(eventtime, status) ~ strata(X1),
  data = d, y = TRUE, x = TRUE)
p.cox <- predictCox(m.cox, newdata = newdata, time = Utime, type = "survival")
p.coxStrata <- predictCox(mStrata.cox, newdata = newdata, time = Utime, type = "survival")

## compare models
outIF <- influenceTest(list(m.cox, mStrata.cox),
  type = "survival", newdata = newdata, times = Utime[1:6])
confint(outIF)

#### Under H0 (CSC) ####
set.seed(1)
ff <- ~ f(X1,2) + f(X2,-0.033)
ff <- update(ff, ~ .+ f(X3,0) + f(X4,0) + f(X5,0))
ff <- update(ff, ~ .+ f(X6,0) + f(X7,0) + f(X8,0) + f(X9,0))
d <- sampleData(n, outcome = "competing.risk", formula = ff)
d[,X1:=as.numeric(as.character(X1))]
d[,X2:=as.numeric(as.character(X2))]
d[,X3:=as.numeric(as.character(X3))]
d[,X4:=as.numeric(as.character(X4))]
d[,X5:=as.numeric(as.character(X5))]
IPA

setkey(d, time)

Utime <- sort(unique(d$time))

## fit cox models
m.CSC <- CSC(Hist(time, event) ~ X1 + X2, data = d)
m.Strata.CSC <- CSC(Hist(time, event) ~ strata(X1) + X2 + X3, data = d)

## compare models
outIF <- influenceTest(list(m.CSC, m.Strata.CSC),
                         cause = 1, newdata = unique(d[,.(X1,X2,X3)]), times = Utime[1:5])
confint(outIF)

---

**IPA**

*Explained variation for settings with binary, survival and competing risk outcome*

### Description
Index of Prediction Accuracy: General $R^2$ for binary outcome and right censored time to event (survival) outcome also with competing risks

### Usage
```r
rsquared(object,...)
IPA(object,...)
## Default S3 method:
rsquared(object,formula,newdata,times,cause,...)
## S3 method for class 'glm'
rsquared(object,formula,newdata,...)
## S3 method for class 'coxph'
rsquared(object,formula,newdata,times,...)
## S3 method for class 'CauseSpecificCox'
rsquared(object,formula,newdata,times,cause,...)
## Default S3 method:
IPA(object,formula,newdata,times,cause,...)
## S3 method for class 'glm'
IPA(object,formula,newdata,...)
## S3 method for class 'coxph'
IPA(object,formula,newdata,times,...)
## S3 method for class 'CauseSpecificCox'
IPA(object,formula,newdata,times,cause,...)
```

### Arguments
- `object`: Model for which we want IPA.
- `...`: passed to `riskRegression::Score`
- `newdata`: Optional validation data set in which to compute IPA
IPA (R^2) is calculated based on the model’s predicted risks. The Brier score of the model is compared to the Brier score of the null model.

Details
IPA (R^2) is calculated based on the model’s predicted risks. The Brier score of the model is compared to the Brier score of the null model.

Value
Data frame with explained variation values for the full model.

Author(s)
Thomas A. Gerds <tag@biostat.ku.dk>

See Also
Score

Examples
```r
library(prodlim)
library(data.table)
# binary outcome
library(lava)
set.seed(18)
learndat <- sampleData(48, outcome="binary")
lr1 = glm(Y~X1+X2+X7+X9, data=learndat, family=binomial)
IPA(lr1)

## validation data
valdat=sampleData(94, outcome="binary")
IPA(lr1, newdata=valdat)

## predicted risks externally given
p1=predictRisk(lr1, newdata=valdat)
IPA(p1, formula=Y~1, valdat)

# survival
library(survival)
data(pbc)
pbc=na.omit(pbc)
pbctest=(1:NROW(pbc)) %in% sample(1: NROW(pbc), size=.632*NROW(pbc))
pbclearn=pbc[pbctest,]
cox1= coxph(Surv(time,status!=0)- age+sex+log(bili)+log(albumin)+log(protime),
            data=pbclearn, x=TRUE)
```
```r
## same data
IPA(cox1, formula = Surv(time, status != 0) ~ 1, times = 1000)

## validation data
pbcval = pbc[!pbctest,]
IPA(cox1, formula = Surv(time, status != 0) ~ 1, newdata = pbcval, times = 1000)

## predicted risks externally given
p2 = predictRisk(cox1, newdata = pbcval, times = 1000)
IPA(cox1, formula = Surv(time, status != 0) ~ 1, newdata = pbcval, times = 1000)

# competing risks
data(Melanoma)
Melanomatest = (1: NROW(Melanoma)) %in% sample(1: NROW(Melanoma), size = .632 * NROW(Melanoma))
Melanomalearn = Melanoma[Melanomatest,]
fit1 <- CSC(list(Hist(time, status) ~ sex,
                Hist(time, status) ~ invasion + epicel + age),
               data = Melanoma)
IPA(fit1, times = 1000, cause = 2)

## validation data
Melanomaval = Melanoma[!Melanomatest,]
IPA(fit1, formula = Hist(time, status) ~ 1, newdata = Melanomaval, times = 1000)

## predicted risks externally given
p3 = predictRisk(fit1, cause = 1, newdata = Melanomaval, times = 1000)
IPA(p3, formula = Hist(time, status) ~ 1, cause = 1, newdata = Melanomaval, times = 1000)
```

---

**ipcw**

*Estimation of censoring probabilities*

**Description**

This function is used internally to obtain inverse of the probability of censoring weights.

**Usage**

```r
ipcw(
  formula,
  data,
  method,
  args,
  times,
  subject.times,
  lag = 1,
  what,
  keep = NULL
)
```
Arguments

formula
A survival formula like, Surv(time, status)~1, where as usual status=0 means censored. The status variable is internally reversed for estimation of censoring rather than survival probabilities. Some of the available models (see argument model) will use predictors on the right hand side of the formula.

data
The data used for fitting the censoring model

method
Censoring model used for estimation of the (conditional) censoring distribution.

args
A list of arguments which is passed to method

times
For what="IPCW.times" a vector of times at which to compute the probabilities of not being censored.

subject.times
For what="IPCW.subject.times" a vector of individual times at which the probabilities of not being censored are computed.

lag
If equal to 1 then obtain G(T_i-|X_i), if equal to 0 estimate the conditional censoring distribution at the subject.times, i.e. G(T_i|X_i).

what
Decide about what to do: If equal to "IPCW.times" then weights are estimated at given times. If equal to "IPCW.subject.times" then weights are estimated at individual subject.times. If missing then produce both.

keep
Which elements to add to the output. Any subset of the vector c("times","fit","call").

Details

Inverse of the probability of censoring weights (IPCW) usually refer to the probabilities of not being censored at certain time points. These probabilities are also the values of the conditional survival function of the censoring time given covariates. The function ipcw estimates the conditional survival function of the censoring times and derives the weights.

IMPORTANT: the data set should be ordered, order(time,-status) in order to get the values IPCW.subject.times in the right order for some choices of method.

Value

A list with elements depending on argument keep.

times
The times at which weights are estimated

IPCW.times
Estimated weights at times

IPCW.subject.times
Estimated weights at individual time values subject.times

fit
The fitted censoring model

method
The method for modelling the censoring distribution

call
The call

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>
Examples

```r
library(prodlim)
library(rms)
dat=SimSurv(30)

dat <- dat[order(dat$time),]

# using the marginal Kaplan-Meier for the censoring times

WKM=ipcw(Hist(time,status)~X2,
data=dat,
method="marginal",
times=sort(unique(dat$time)),
subject.times=dat$time,keep=c("fit"))
plot(WKM$fit)
WKM$fit

# using the Cox model for the censoring times given X2
library(survival)
WCox=ipcw(Hist(time=time,event=status)~X2,
data=dat,
method="cox",
times=sort(unique(dat$time)),
subject.times=dat$time,keep=c("fit"))
WCox$fit

plot(WKM$fit)
lines(sort(unique(dat$time)),
1-WCox$IPCW.times[1,],
type="1",
col=2,
lty=3,
lwd=3)
lines(sort(unique(dat$time)),
1-WCox$IPCW.times[5,],
type="1",
col=3,
lty=3,
lwd=3)

# using the stratified Kaplan-Meier
# for the censoring times given X2

WKM2=ipcw(Hist(time,status)~X2,
data=dat,
method="nonpar",
times=sort(unique(dat$time)),
subject.times=dat$time,keep=c("fit"))
plot(WKM2$fit,add=FALSE)
```
Description

In the period 1962-77, 205 patients with malignant melanoma (cancer of the skin) had a radical operation performed at Odense University Hospital, Denmark. All patients were followed until the end of 1977 by which time 134 were still alive while 71 had died (of out whom 57 had died from cancer and 14 from other causes).

Format

A data frame with 205 observations on the following 12 variables.

- **time**: time in days from operation
- **status**: a numeric with values 0=censored 1=death.malignant.melanoma 2=death.other.causes
- **event**: a factor with levels censored death.malignant.melanoma death.other.causes
- **invasion**: a factor with levels level.0, level.1, level.2
- **ici**: inflammatory cell infiltration (IFI): 0, 1, 2 or 3
- **epicel**: a factor with levels not present present
- **ulcer**: a factor with levels not present present
- **thick**: tumour thickness (in 1/100 mm)
- **sex**: a factor with levels Female Male
- **age**: age at operation (years)
- **logthick**: tumour thickness on log-scale

Details

The object of the study was to assess the effect of risk factors on survival. Among such risk factors were the sex and age of the patients and the histological variables tumor thickness and ulceration (absent vs. present).

References

Regression with linear predictors (2010)
Andersen, P.K. and Skovgaard, L.T.
Springer Verlag

Examples

data(Melanoma)
model.matrix.cph  Extract design matrix for cph objects

Description

Extract design matrix for cph objects

Usage

## S3 method for class 'cph'
model.matrix(object, data)

Arguments

- object: a cph object.
- data: a dataset.

model.matrix.phreg  Extract design matrix for phreg objects

Description

Extract design matrix for phreg objects

Usage

## S3 method for class 'phreg'
model.matrix(object, data)

Arguments

- object: a phreg object.
- data: a dataset.

Details

mainly a copy paste of the begining of the phreg function.
Paquid

Description

PAQUID is a prospective cohort study initiated in 1988 in South Western France to explore functional and cerebral ageing. This sample includes n=2561 subjects. Data contains a time-to-event, a type of event and two cognitive scores measured at baseline.

Format

A data frame with 2561 observations on the following 4 variables.

- **time**: the time-to-event (in years).
- **status**: the type of event $0 = \text{censored}$, $1 = \text{dementia onset}$ and $2 = \text{death without dementia}$.
- **DSST**: score at the Digit Symbol Substitution Score Test. This test explores attention and psychomotor speed.
- **MMSE**: score at the Mini Mental State Examination. This test is often used as an index of global cognitive performance.

Source

The data have been first made publicly available via the package timeROC.

References


Examples

data(Paquid)
penalizedS3  

S3-wrapper for S4 function penalized

Description

S3-wrapper for S4 function penalized

Usage

penalizedS3(formula, data, type = "elastic.net", lambda1, lambda2, fold, ...)

Arguments

formula  Communicated outcome and explanatory variables. See examples.
data  Data set in which formula is to be interpreted
type  String specifying the type of penalization. Should match one of the following values: "ridge", "lasso", "elastic.net".
lambda1  Lasso penalty
lambda2  ridge penalty
fold  passed to penalized::profL1
...
Arguments passed to penalized

Examples

library(prodlim)
## Not run:
## too slow
library(penalized)
set.seed(8)
d <- sampleData(200, outcome="binary")
newd <- sampleData(80, outcome="binary")
fitridge <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="ridge", standardize=TRUE, model="logistic", trace=FALSE)
fitlasso <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="lasso", standardize=TRUE, model="logistic", trace=FALSE)
# fitnet <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="elastic.net", standardize=TRUE, model="logistic", trace=FALSE)
predictRisk(fitridge, newdata=newd)
predictRisk(fitlasso, newdata=newd)
# predictRisk(fitnet, newdata=newd)
Score(list(fitridge), data=newd, formula=Y~1)
Score(list(fitridge), data=newd, formula=Y~1, split.method="bootcv", B=2)

## End(Not run)
## Not run: data(nki70) ## S4 fit
pen <- penalized(Surv(time, event), penalized = nki70[,8:77], unpenalized = ~ER+Age+Diam+N+Grade, data = nki70,
## S3 method for class 'riskRegression'
plot(x, 
cause, 
newdata, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
...)

plot.riskRegression  

**Description**

Show predicted risk obtained by a risk prediction model as a function of time.

**Usage**

```r
## S3 method for class 'riskRegression'
plot(x, 
cause, 
newdata, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
lab, 
...)
```

**Arguments**

- `x`  
  Fitted object obtained with one of ARR, LRR, riskRegression.

- `cause`  
  For CauseSpecificCox models the cause of interest.

- `newdata`  
  A data frame containing predictor variable combinations for which to compute predicted risk.

- `lab`  
  See plot

- `lab`  
  See plot

- `lab`  
  See plot
plotAUC

Plot of time-dependent AUC curves

Description

Plot of time-dependent AUC curves

Usage

plotAUC(
  x,
  models,
  which = "score",
  xlim, ylim, xlab, ylab, col, lwd,
  lty = 1,
  ylim
  lwd
  col
  lty
  axes
  percent
  legend
  add
  ...
)

Arguments

xlim
See plot
lwd A vector of line thicknesses for the regression coefficients.
col A vector of colors for the regression coefficients.
lty A vector of line types for the regression coefficients.
axes Logical. If FALSE then do not draw axes.
percent If true the y-axis is labeled in percent.
legend If true draw a legend.
add Logical. If TRUE then add lines to an existing plot.
... Used for transclusion of smart arguments for plot, lines, axis and background.

See function SmartControl from prodlim.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

library(survival)
library(prodlim)
data(Melanoma)
fit.arr <- ARR(Hist(time,status)~invasion+age+strata(sex),data=Melanoma,cause=1)
plot(fit.arr,xlim=c(500,3000))
plotAUC

cex = 1,
pch = 1,
type = "l",
axes = 1L,
percent = 1L,
conf.int = 0L,
legend = 1L,
...
)

Arguments

x Object obtained with Score.list
models Choice of models to plot
which Character. Either "score" to show AUC or "contrasts" to show differences between AUC.
xlim Limits for x-axis
ylim Limits for y-axis
xlab Label for x-axis
ylab Label for y-axis
col line color
lwd line width
lty line style
cex point size
pch point style
type line type
axes Logical. If TRUE draw axes.
percent Logical. If TRUE scale y-axis in percent.
conf.int Logical. If TRUE draw confidence shadows.
legend Logical. If TRUE draw legend.
... Used for additional control of the subroutines: plot,

Examples

library(survival)
library(prodlim)
d=sampleData(100,outcome="survival")
nd=sampleData(100,outcome="survival")
f1=coxph(Surv(time,event)~X1+X6+X8,data=d,x=TRUE,y=TRUE)
f2=coxph(Surv(time,event)~X2+X5+X9,data=d,x=TRUE,y=TRUE)
xx=Score(list("X1+X6+X8"=f1,"X2+X5+X9"=f2), formula=Surv(time,event)~1,
data=nd, metrics="auc", null.model=FALSE, times=seq(3:10))
aucgraph <- plotAUC(xx)
plotAUC(xx,conf.int=TRUE)
## difference between
plotBrier

plotAUC(xx, which="contrasts", conf.int=TRUE)

---

plotBrier  Plot Brier curve

Description

Plot Brier score curves

Usage

plotBrier(
  x,
  models,
  which = "score",
  xlim,
  ylim,
  xlab,
  ylab,
  col,
  lwd,
  lty = 1,
  cex = 1,
  pch = 1,
  type = "l",
  axes = 1L,
  percent = 1L,
  conf.int = 0L,
  legend = 1L,
  ...)

Arguments

  x          Object obtained with Score
  models     Choice of models to plot
  which      Character. Either "score" to show AUC or "contrasts" to show differences
             between AUC.
  xlim       Limits for x-axis
  ylim       Limits for y-axis
  xlab       Label for x-axis
  ylab       Label for y-axis
  col        line color
lwd line width
lty line style
cex point size
pch point style
type line type
axes Logical. If TRUE draw axes.
percent Logical. If TRUE scale y-axis in percent.
conf.int Logical. If TRUE draw confidence shadows.
legend Logical. If TRUE draw legend.
... Used for additional control of the subroutines: plot, axis, lines, legend. See SmartControl.

Examples

# survival
library(survival)
library(prodlim)
ds1=sampleData(40.outcome="survival")
ds2=sampleData(40.outcome="survival")
f1 <- coxph(Surv(time,event)-X1+X3+X5+X7+X9,data=ds1,x=TRUE)
f2 <- coxph(Surv(time,event)-X2+X4+X6+X8+X10,data=ds1,x=TRUE)
xscore <- Score(list(f1,f2),formula=Hist(time,event)-1,data=ds2,times=0:12,metrics="brier")
plotBrier(xscore)

plotCalibration Plot Calibration curve

Description

Plot Calibration curve

Usage

plotCalibration(
x, models, times,
method = "nne", cens.method,
round = TRUE, bandwidth = NULL,
q = 10, bars = FALSE, hanging = FALSE, names = "quantiles",
pseudo = FALSE,
rug,
show.frequencies = FALSE,
plot = TRUE,
add = FALSE,
diag = !add,
legend = !add,
 auc.in.legend,
brier.in.legend,
axes = !add,
xlim = c(0, 1),
ylim = c(0, 1),
xlab = ifelse(bars, "Risk groups", "Predicted risk"),
ylab = "Observed frequency",
col,
lwd,
lty,
pch,
type,
percent = TRUE,
auc.plot = FALSE,
na.action = na.fail,
cex = 1,
...)

Arguments

x Object obtained with function Score
models Choice of models to plot
times Time point specifying the prediction horizon.
method The method for estimating the calibration curve(s):

• "quantile" The observed proportion at predicted risk value 'p' is obtained in groups defined by quantiles of the predicted event probabilities of all subjects. The number of groups is controlled by argument q.

• "nne" The observed proportion at predicted risk value 'p' is obtained based on the subjects whose predicted risk is inside a nearest neighbor hood around the value 'p'. The larger the bandwidth the more subjects are included in the current neighborhood.

cens.method For right censored data only. How observed proportions are calculated. Either "jackknife" or "local":

• "jackknife" Compute a running mean of the jackknife pseudovalues across neighborhoods/groups of the predicted risks. Here we rely on the assumption that censoring is independent of the event time and the covariates, see References.

• "local" Compute the Kaplan-Meier estimator in absence of competing risks and the Aalen-Johansen estimator in presence of competing risks locally
like a running mean in neighborhoods of the predicted risks. The widths of
the neighborhoods are defined according to method.

round If TRUE predicted probabilities are rounded to two digits before smoothing. This
may have a considerable effect on computing efficiency in large data sets.

bandwidth The bandwidth for method="nne"

q The number of quantiles for method="quantile" and bars=TRUE.

bars If TRUE, use barplots to show calibration.

hanging Barplots only. If TRUE, hang bars corresponding to observed frequencies at the
value of the corresponding prediction.

names Barplots only. Names argument passed to names.arg of barplot.

pseudo If TRUE show pseudo values (only for right censored data).

rug If TRUE show rug plot at the predictions

show.frequencies Barplots only. If TRUE, show frequencies above the bars.

plot If FALSE, do not plot the results, just return a plottable object.

add If TRUE the line(s) are added to an existing plot.

diag If FALSE no diagonal line is drawn.

legend Logical. If TRUE draw legend.

auc.in.legend Logical. If TRUE add AUC to legend.

brier.in.legend Logical. If TRUE add Brier score to legend.

axes If FALSE no axes are drawn.

xlim Limits of x-axis.

ylim Limits of y-axis.

xlab Label for y-axis.

ylab Label for x-axis.

col Vector with colors, one for each element of object. Passed to lines.

lwd Vector with line widths, one for each element of object. Passed to lines.

lty lwd Vector with line style, one for each element of object. Passed to lines.

pch Passed to lines.

type Passed to lines.

percent If TRUE axes labels are multiplied by 100 and thus interpretable on a percent
scale.

na.action what to do with NA values. Passed to model.frame

cex Default cex used for legend and labels.

... Used to control the subroutines: plot, axis, lines, barplot, legend, addtable2plot,
points (pseudo values), rug. See SmartControl.
plotEffects

Examples

```r
library(prodlim)

# binary
db=sampleData(100,outcome="binary")
fb1=glm(Y~X1+X5+X7,data=db,family="binomial")
fb2=glm(Y~X1+X3+X6+X7,data=db,family="binomial")
xb=Score(list(model1=fb1,model2=fb2),Y~1,data=db,
    plots="cal")
plotCalibration(xb,brier.in.legend=TRUE)
plotCalibration(xb,bars=TRUE,model="model1")
plotCalibration(xb,models=1,bars=TRUE,names.cex=1.3)

# survival
library(survival)
library(prodlim)
dslearn=sampleData(56,outcome="survival")
dtest=sampleData(100,outcome="survival")
fs1=coxph(Surv(time,event)-X1+X5+X7,data=dlearn,x=1)
fs2=coxph(Surv(time,event)-strata(X1)+X3+X6+X7,data=dlearn,x=1)
x=Score(list(Cox1=fs1,Cox2=fs2),Surv(time,event)-1,data=dtest,
    plots="cal",metrics=NULL)
plotCalibration(x)
plotCalibration(x,cens.method="local",pseudo=1)
plotCalibration(x,method="quantile")

# competing risks
## Not run:
data(Melanoma)
f1 <- CSC(Hist(time,status)-age+sex+epicel+ulcer,data=Melanoma)
f2 <- CSC(Hist(time,status)-age+sex+logthick+epicel+ulcer,data=Melanoma)
x <- Score(list(model1=f1,model2=f2),Hist(time,status)-1,data=Melanoma,
    cause= 2,times=5*365.25,plots="cal")
plotCalibration(x)
## End(Not run)
```

Description

Plotting time-varying effects from a risk regression model.

Usage

```r
plotEffects(
```

plotEffects

Plotting time-varying effects from a risk regression model.
plotEffects

Arguments

x Fitted object obtained with one of ARR, LRR, riskRegression.
formula A formula to specify the variable(s) whose regression coefficients should be plotted.
level For categorical variables the level (group) whose contrast to the reference level (group) should be plotted.
ref.line Logical. If TRUE then add a horizontal line at zero.
conf.int Logical. If TRUE then add confidence limits. Can be controlled using smart arguments. See examples
xlim See plot
ylim See plot
xlab See plot
ylab See plot
col A vector of colors for the regression coefficients.
lty A vector of line types for the regression coefficients.
lwd A vector of line thicknesses for the regression coefficients.
add Logical. If TRUE then add lines to an existing plot.
legend Logical. If TRUE then add a legend. Can be controlled using smart arguments. See examples.
axes Logical. If FALSE then do not draw axes.
... Used for transclusion of smart arguments for plot, axis. See function SmartControl from prodlim.

Author(s)

Thomas H. Scheike <ts@biostat.ku.dk>
Thomas A. Gerds <tag@biostat.ku.dk>
Examples

```r
library(survival)
library(prodlim)
data(Melanoma)

fit.tarr <- ARR(Hist(time,status)~strata(sex),
data=Melanoma,
cause=1)
plotEffects(fit.tarr)

fit.tarr <- ARR(Hist(time,status)~strata(sex)+strata(invasion),
data=Melanoma,
cause=1,
times=seq(800,3000,20))
plotEffects(fit.tarr,formula=~sex)
plotEffects(fit.tarr,formula=~invasion)
plotEffects(fit.tarr,
  formula=~invasion,
  level="invasionlevel.1")
```

## legend arguments are transcluded:
```r
plotEffects(fit.tarr,
  formula=~invasion,
  legend.bty="b",
  legend.cex=1)
```

## and other smart arguments too:
```r
plotEffects(fit.tarr,
  formula=~invasion,
  legend.bty="b",
  axis2.las=2,
  legend.cex=1)
```

---

**plotPredictRisk**

Plotting predicted risks curves.

**Description**

Time-dependent event risk predictions.

**Usage**

```r
plotPredictRisk(
x,
newdata,
times,
```
 cause = 1,
xlim, ylim, xlab, ylab, axes = TRUE, col, density, lty, lwd, add = FALSE, legend = TRUE, percent = FALSE,
... )

Arguments

x Object specifying an event risk prediction model.
newdata A data frame with the same variable names as those that were used to fit the model x.
times Vector of times at which to return the estimated probabilities.
cause Show predicted risk of events of this cause
xlim Plotting range on the x-axis.
ylim Plotting range on the y-axis.
xlab Label given to the x-axis.
ylab Label given to the y-axis.
axes Logical. If FALSE no axes are drawn.
col Vector of colors given to the survival curve.
density Density of the color – useful for showing many (overlapping) curves.
lty Vector of lty’s given to the survival curve.
lwd Vector of lwd’s given to the survival curve.
add Logical. If TRUE only lines are added to an existing device
legend Logical. If TRUE a legend is plotted by calling the function legend. Optional arguments of the function legend can be given in the form legend.x=val where x is the name of the argument and val the desired value. See also Details.
percent Logical. If TRUE the y-axis is labeled in percent.
... Parameters that are filtered by SmartControl and then passed to the functions: plot, axis, legend.
Details

Arguments for the invoked functions legend and axis can be specified as legend.lty=2. The specification is not case sensitive, thus Legend.lty=2 or LEGEND.lty=2 will have the same effect. The function axis is called twice, and arguments of the form axis1.labels, axis1.at are used for the time axis whereas axis2.pos, axis1.labels, etc., are used for the y-axis.

These arguments are processed via ...{} of plotPredictRisk and inside by using the function SmartControl.

Value

The (invisible) object.

Author(s)

Ulla B. Mogensen and Thomas A. Gerds <tag@biostat.ku.dk>

References


See Also

plotRisk

Examples

library(survival)
# generate survival data
# no effect
set.seed(8)
d <- sampleData(80,outcome="survival",formula = ~f(X6, 0) + f(X7, 0))
d[,table(event)]
f <- coxph(Surv(time,event)~X6+X7,data=d,x=1)
plotPredictRisk(f)

# large effect
set.seed(8)
d <- sampleData(80,outcome="survival",formula = ~f(X6, 0.1) + f(X7, -0.1))
d[,table(event)]
f <- coxph(Surv(time,event)~X6+X7,data=d,x=1)
plotPredictRisk(f)

# generate competing risk data
# small effect
set.seed(8)
d <- sampleData(40,formula = ~f(X6, 0.01) + f(X7, -0.01))
d[,table(event)]
f <- CSC(Hist(time,event)~X5+X6,data=d)
plotPredictRisk(f)
```r
# large effect
set.seed(8)
d <- sampleData(40, formula = ~f(X6, 0.1) + f(X7, -0.1))
d[, table(event)]
f <- CSC(Hist(time, event) ~ X5 + X6, data = d)
plotPredictRisk(f)
```

---

**plotRisk**  
*plot predicted risks*

**Description**

plot predicted risks

**Usage**

```r
plotRisk(
  x,  
  models,  
  times,  
  xlim = c(0, 1),  
  ylim = c(0, 1),  
  xlab,  
  ylab,  
  col,  
  pch,  
  cex = 1,  
  preclipse = 0,  
  preclipse.shade = FALSE,  
  ...
)
```

**Arguments**

- `x` Object obtained with function `Score`
- `models` Choice of two models to plot. The predicted risks of the first (second) are shown along the x-axis (y-axis).
- `times` Time point specifying the prediction horizon.
- `xlim` x-axis limits
- `ylim` y-axis limits
- `xlab` x-axis labels
- `ylab` y-axis labels
- `col` Colors used according to the outcome. binary outcome (two colors: no event, event), survival outcome (three colors: censored, event, no event) competing risk outcome (4 or more colors: event, competing risk 1, ..., competing risk k, censored, no event)
Symbols used according to the outcome

- **binary outcome** (two symbols: no event, event)
- **survival outcome** (three symbols: censored, event, no event)
- **competing risk outcome** (4 or more symbols: event, competing risk 1, ..., competing risk k, censored, no event)

**pch**
Point size

**preclipse**
Value between 0 and 1 defining the preclipse area

**preclipse.shade**
Logical. If TRUE shade the area of clinically meaningful change.

Details

Two rival prediction models are applied to the same data.

Value

A nice graph

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```r
library(prodlim)
## uncensored
learndat = sampleData(40,outcome="binary")
testdat = sampleData(40,outcome="binary")
lr1 = glm(Y~X1+X2+X7+X9,data=learndat,family="binomial")
lr2 = glm(Y~X3+X5+X6,data=learndat,family="binomial")
xb=Score(list("LR(X1+X2+X7+X9)"=lr1,"LR(X3+X5+X6)"=lr2),formula=Y~1,
data=testdat,summary="risks",null.model=0L)
plotRisk(xb)
## survival
library(survival)
learndat = sampleData(40,outcome="survival")
testdat = sampleData(40,outcome="survival")
cox1 = coxph(Surv(time,event)~X1+X2+X7+X9,data=learndat,x=TRUE)
cox2 = coxph(Surv(time,event)~X3+X5+X6,data=learndat,x=TRUE)
xs=Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),formula=Surv(time,event)-1,
data=testdat,summary="risks",null.model=0L,times=c(3,5,6))
plotRisk(xs,times=5)
## competing risk
library(prodlim)
library(survival)
set.seed(8)
learndat = sampleData(80,outcome="competing.risk")
testdat = sampleData(140,outcome="competing.risk")
m1 = FGR(Hist(time,event)~X2+X7+X9,data=learndat,cause=1)
```

...
m2 = CSC(Hist(time,event)~X2+X7+X9,data=learndat,cause=1)
xcr=Score(list("FGR"=m1,"CSC"=m2),formula=Hist(time,event)=1,
  data=testdat,summary="risks",null.model=0L,times=c(3,5))
plotRisk(xcr,times=1)
## End(Not run)

---

### plotROC

**Plot ROC curves**

#### Description

Plot ROC curve

#### Usage

```r
plotROC(
  x,
  models,
  times,
  xlab = "1-Specificity",
  ylab = "Sensitivity",
  col,
  lwd,
  lty = 1,
  cex = 1,
  pch = 1,
  legend = TRUE,
  auc.in.legend = TRUE,
  brier.in.legend = FALSE,
  add = FALSE,
  ...
)
```

#### Arguments

- **x**: Object obtained with function `Score`
- **models**: Choice of models to plot
- **times**: Time point(s) specifying the prediction horizon
- **xlab**: Label for x-axis
- **ylab**: Label for y-axis
- **col**: line color
- **lwd**: line width
- **lty**: line style
- **cex**: point size
plotROC

pch         point style
legend      logical. If 1L draw a legend with the values of AUC.
auc.in.legend Logical. If TRUE add AUC to legend.
brier.in.legend Logical. If TRUE add Brier score to legend.
add         logical. If 1L add lines to an existing plot.
...

Used for additional control of the subroutines: plot, axis, lines, legend, addtable2plot. See SmartControl.

Examples

## binary
set.seed(18)
library(randomForest)
library(prodlim)
bd1 <- sampleData(40,outcome="binary")
bd2 <- sampleData(58,outcome="binary")
bd1[,y:=factor(Y)]
bd2[,y:=factor(Y)]
fb1 <- glm(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10,data=bd1,family="binomial")
fb2 <- randomForest(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10,data=bd1)
xb <- Score(list("glm"=fb1,"rf"=fb2),y~1,data=bd2,
            plots="roc",metrics=c("auc","brier"))
plotROC(xb,brier.in.legend=1L)

# with cross-validation
## Not run:
xb3 <- Score(list("glm"=fb1,"rf"=fb2),y~1,data=bd1,
             plots="roc",B=3,split.method="bootcv",
             metrics=c("auc"))

## End(Not run)

## survival
set.seed(18)
library(survival)
sdl <- sampleData(40,outcome="survival")
sdt <- sampleData(58,outcome="survival")
sf1 <- coxph(Surv(time,event)~X3+X5+X6+X7+X8+X10,data=sdl,x=TRUE)
sf2 <- coxph(Surv(time,event)~X1+X2+X9,data=sdl,x=TRUE)
xs <- Score(list(model1=f1,model2=f2),Hist(time,event)~1,data=sdt,
            times=5,plots="roc",metrics="auc")
plotROC(xs)

## competing risks
data(Melanoma)
f1 <- CSC(Hist(time,status)~age+sex+epicel+ulcer,data=Melanoma)
f2 <- CSC(Hist(time,status)~age+sex+logthick+epicel+ulcer,data=Melanoma)
x <- Score(list(model1=f1,model2=f2),Hist(time,status)~1,data=Melanoma,
          cause=1,times=5*365.25,plots="roc",metrics="auc")
plotROC(x)
**predict.CauseSpecificCox**

*Predicting Absolute Risk from Cause-Specific Cox Models*

**Description**

Apply formula to combine two or more Cox models into absolute risk (cumulative incidence function).

**Usage**

```r
## S3 method for class 'CauseSpecificCox'
predict(
  object,
  newdata,  # data.frame or data.table Contain the values of the predictor variables defining subject specific predictions relative to each cause. Should have the same structure as the data set used to fit the object.
  times,  # numeric vector Time points at which to return the estimated absolute risk.
  cause,  # integer/character Identifies the cause of interest among the competing events.
  type = "absRisk",  # character Can be changed to "survival" if the event free survival should be output instead of the absolute risk.
  landmark = NA,  # integer The starting time for the computation of the cumulative risk.
  keep.times = 1L,  # logical If TRUE add the evaluation times to the output.
  keep.newdata = 1L,
  keep.strata = 1L,
  se = FALSE,
  band = FALSE,
  iid = FALSE,
  confint = (se + band) > 0,
  average.iid = FALSE,
  product.limit = TRUE,
  store.iid = "full",
  diag = FALSE,
  ...)
```

**Arguments**

- **object**: The fitted cause specific Cox model.
- **newdata**: data.frame or data.table Contain the values of the predictor variables defining subject specific predictions relative to each cause. Should have the same structure as the data set used to fit the object.
- **times**: numeric vector Time points at which to return the estimated absolute risk.
- **cause**: integer/character Identifies the cause of interest among the competing events.
- **type**: character Can be changed to "survival" if the event free survival should be output instead of the absolute risk.
- **landmark**: integer The starting time for the computation of the cumulative risk.
- **keep.times**: logical If TRUE add the evaluation times to the output.
`predict.CauseSpecificCox`

```r
default predict.CauseSpecificCox

```predict.CauseSpecificCox parameters:

- `keep.newdata` [logical] If TRUE add the value of the covariates used to make the prediction in the output list.
- `keep.strata` [logical] If TRUE add the value of the strata used to make the prediction in the output list.
- `se` [logical] If TRUE compute and add the standard errors to the output.
- `band` [logical] If TRUE compute and add the quantiles for the confidence bands to the output.
- `iid` [logical] If TRUE compute and add the influence function to the output.
- `confint` [logical] If TRUE compute and add the confidence intervals/bands to the output. They are computed applying the `confint` function to the output.
- `average.iid` [logical]. If TRUE add the average of the influence function over `newdata` to the output.
- `product.limit` [logical]. If TRUE the survival is computed using the product limit estimator. Otherwise the exponential approximation is used (i.e. `exp(-cumulative hazard)`).
- `store.iid` [character] Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal".
- `diag` [logical] when FALSE the absolute risk/survival for all observations at all times is computed, otherwise it is only computed for the i-th observation at the i-th time.

... not used.

Details

This function computes the absolute risk as given by formula 2 of (Ozenne et al., 2017). Confidence intervals and confidence bands can be computed using a first order von Mises expansion. See the section "Construction of the confidence intervals" in (Ozenne et al., 2017).

A detailed explanation about the meaning of the argument `store.iid` can be found in (Ozenne et al., 2017) Appendix B "Saving the influence functions".

Note: for Cox regression models with time varying covariates it does not make sense to use this function, because the predicted risk has to be a measurable function of the data available at the time origin.

The iid decomposition is output using an array containing the value of the influence of each subject used to fit the object (dim 3), for each subject in newdata (dim 1), and each time (dim 2).

Author(s)

Brice Ozenne broz@sund.ku.dk, Thomas A. Gerds tag@biostat.ku.dk

References

See Also

confint.predictCSC to compute confidence intervals/bands. autoplot.predictCSC to display the predictions.

Examples

library(survival)
library(prodlim)
### generate data ###
set.seed(5)
d <- sampleData(80, outcome="comp") ## training dataset
nd <- sampleData(4, outcome="comp") ## validation dataset
d$time <- round(d$time, 1) ## create tied events

## estimate a CSC model based on the coxph function
CSC.fit <- CSC(Hist(time, event) ~ X3 + X8, data = d, method = "breslow")

## compute the absolute risk of cause 1, in the validation dataset
## at time 1:10
CSC.risk <- predict(CSC.fit, newdata = nd, times = 1:10, cause = 1)
CSC.risk

## compute absolute risks with CI for cause 2
## (without displaying the value of the covariates)
predict(CSC.fit, newdata = nd, times = 1:10, cause = 2, se = TRUE,
keep.newdata = FALSE)

## other example
library(survival)
CSC.fit.s <- CSC(list(Hist(time, event) ~ strata(X1) + X2 + X9,
Hist(time, event) ~ X2 + strata(X4) + X8 + X7), data = d, method = "breslow")
predict(CSC.fit.s, cause = 1, times = ttt, se = 1L) ## note: absRisk>1 due to small number of observations

## using the cph function instead of coxph
CSC.cph <- CSC(Hist(time, event) ~ X1 + X2, data = d, method = "breslow", fitter = "cph")#
predict(CSC.cph, newdata = d, cause = 2, times = ttt)

## landmark analysis
T0 <- 1
predCSC_afterT0 <- predict(CSC.fit, newdata = d, cause = 2, times = ttt[ttt>T0], landmark = T0)
predCSC_afterT0

---

predict.FGR

Predict subject specific risks (cumulative incidence) based on Fine-Gray regression model

**Description**

Predict subject specific risks (cumulative incidence) based on Fine-Gray regression model
Usage

## S3 method for class 'FGR'
predict(object, newdata, times, ...)

Arguments

object Result of call to FGR
newdata Predictor values of subjects for who to predict risks
times Time points at which to evaluate the risks
...

Examples

library(prodlim)
library(survival)
set.seed(10)
d <- sampleData(101, outcome = "competing.risk")
tFun<-function(t) {t}
fgr<-FGR(Hist(time, event)~X1+strata(X2)+X6+cov2(X7, tf=tFun),
data=d, cause=1)
predictRisk(fgr,times=5,newdata=d[1:10])

Description

Predict individual risk.

Usage

## S3 method for class 'riskRegression'
predict(object, newdata, ...)

Arguments

object Fitted object obtained with one of ARR, LRR, riskRegression.
newdata A data frame containing predictor variable combinations for which to compute predicted risk.
...

Author(s)

Thomas H. Scheike <ts@biostat.ku.dk>
Thomas A. Gerds <tag@biostat.ku.dk>
References


Examples

data(Melanoma)
library(prodlim)
library(survival)

fit.tarr <- ARR(Hist(time,status)~age+invasion+strata(sex),data=Melanoma,cause=1)
predict(fit.tarr,newdata=data.frame(age=48,
       invasion=factor("level.1",
               levels=levels(Melanoma$invasion)),
       sex=factor("Female",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=data.frame(age=48,
       invasion=factor("level.1",
               levels=levels(Melanoma$invasion)),
       sex=factor("Male",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=data.frame(age=c(48,58,68),
       invasion=factor("level.1",
               levels=levels(Melanoma$invasion)),
       sex=factor("Male",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=Melanoma[1:4,])

predictCox

Fast computation of survival probabilities, hazards and cumulative hazards from Cox regression models

Description

Fast routine to get baseline hazards and subject specific hazards as well as survival probabilities from a survival::coxph or rms::cph object

Usage

predictCox(
  object,
  times,
  newdata = NULL,
  centered = TRUE,
  type = c("cumhazard", "survival"),
  keep.strata = TRUE,
  keep.times = TRUE,
  keep.newdata = FALSE,
predictCox

keep.infoVar = FALSE,
se = FALSE,
band = FALSE,
 iid = FALSE,
confint = (se + band) > 0,
diag = FALSE,
average.iid = FALSE,
store.iid = "full"
)

Arguments

object       The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
times        [numeric vector] Time points at which to return the estimated hazard/cumulative hazard/survival.
newdata      [data.frame or data.table] Contain the values of the predictor variables defining subject specific predictions. Should have the same structure as the data set used to fit the object.
centered     [logical] If TRUE return prediction at the mean values of the covariates fit$mean, if FALSE return a prediction for all covariates equal to zero. in the linear predictor. Will be ignored if argument newdata is used. For internal use.
type         [character vector] the type of predicted value. Choices are

  • "hazard" the baseline hazard function when argument newdata is not used and the hazard function when argument newdata is used.
  • "cumhazard" the cumulative baseline hazard function when argument newdata is not used and the cumulative hazard function when argument newdata is used.
  • "survival" the survival baseline hazard function when argument newdata is not used and the cumulative hazard function when argument newdata is used.

Several choices can be combined in a vector of strings that match (no matter the case) strings "hazard","cumhazard","survival".

keep.strata  [logical] If TRUE add the (newdata) strata to the output. Only if there any.
keep.times   [logical] If TRUE add the evaluation times to the output.
keep.newdata [logical] If TRUE add the value of the covariates used to make the prediction in the output list.
keep.infoVar [logical] For internal use.
se           [logical] If TRUE compute and add the standard errors to the output.
band         [logical] If TRUE compute and add the quantiles for the confidence bands to the output.
iid          [logical] If TRUE compute and add the influence function to the output.
confint      [logical] If TRUE compute and add the confidence intervals/bands to the output. They are computed applying the confint function to the output.
diag [logical] when FALSE the hazard/cumulative hazard/survival for all observations at all times is computed, otherwise it is only computed for the i-th observation at the i-th time.

average.iid [logical] If TRUE add the average of the influence function over newdata to the output.

store.iid [character] Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal".

Details

When the argument newdata is not specified, the function computes the baseline hazard estimate. See (Ozenne et al., 2017) section "Handling of tied event times".

Otherwise the function computes survival probabilities with confidence intervals/bands. See (Ozenne et al., 2017) section "Confidence intervals and confidence bands for survival probabilities". The survival is computed using the exponential approximation (equation 3).

A detailed explanation about the meaning of the argument store.iid can be found in (Ozenne et al., 2017) Appendix B "Saving the influence functions".

The function is not compatible with time varying predictor variables.

The centered argument enables us to reproduce the results obtained with the basehaz function from the survival package but should not be modified by the user.

The iid decomposition is output using an array containing the value of the influence of each subject used to fit the object (dim 3), for each subject in newdata (dim 1), and each time (dim 2).

Author(s)

Brice Ozenne broz@sund.ku.dk, Thomas A. Gerds tag@biostat.ku.dk

References


See Also

confint.predictCox to compute confidence intervals/bands. autoplot.predictCox to display the predictions.

Examples

library(survival)
library(data.table)

#### generate data ####
set.seed(10)
d <- sampleData(40,outcome="survival") ## training dataset
nd <- sampleData(4,outcome="survival") ## validation dataset
d$time <- round(d$time,1) ## create tied events
# table(duplicated(d$time))

#### stratified Cox model ####
fit <- coxph(Surv(time,event)~X1 + strata(X2) + X6,
    data=d, ties="breslow", x = TRUE, y = TRUE)

## compute the baseline cumulative hazard
fit.haz <- predictCox(fit)
cbind(survival::basehaz(fit), fit.haz$cumhazard)

## compute individual specific cumulative hazard and survival probabilities
fit.pred <- predictCox(fit, newdata=nd, times=c(3,8), se = TRUE, band = TRUE)
fit.pred

#### other examples ####
# one strata variable
fitS <- coxph(Surv(time,event)~strata(X1)+X2,
    data=d, ties="breslow", x = TRUE, y = TRUE)
predictCox(fitS)
predictCox(fitS, newdata=nd, times = 1)

# two strata variables
set.seed(1)
d$U=sample(letters[1:5],replace=TRUE,size=NROW(d))
d$V=sample(letters[4:10],replace=TRUE,size=NROW(d))
nd$U=sample(letters[1:5],replace=TRUE,size=NROW(nd))
nd$V=sample(letters[4:10],replace=TRUE,size=NROW(nd))
fit2S <- coxph(Surv(time,event)~X1+strata(U)+strata(V)+X2,
    data=d, ties="breslow", x = TRUE, y = TRUE)

cbind(survival::basehaz(fit2S),predictCox(fit2S,type="cumhazard")$cumhazard)
predictCox(fit2S)
predictCox(fit2S, newdata=nd, times = 3)

# left truncation
test2 <- list(start=c(1,2,5,2,1,7,3,4,8,8),
    stop=c(2,3,6,7,8,9,9,14,17),
    event=c(1,1,1,1,1,1,1,0,0,0),
    x=c(1,0,0,1,0,1,1,0,0,0))
m.cph <- coxph(Surv(start, stop, event) ~ 1, test2, x = TRUE)
as.data.table(predictCox(m.cph))

basehaz(m.cph)

---

**predictCoxPL**

*Computation of survival probabilities from Cox regression models using the product limit estimator.*
Description

Same as predictCox except that the survival is estimated using the product limit estimator.

Usage

predictCoxPL(
  object,
  times,
  newdata = NULL,
  type = c("cumhazard", "survival"),
  keep.strata = TRUE,
  keep.infoVar = FALSE,
  ...
)

Arguments

- **object**: The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
- **times**: [numeric vector] Time points at which to return the estimated hazard/cumulative hazard/survival.
- **newdata**: [data.frame or data.table] Contain the values of the predictor variables defining subject specific predictions. Should have the same structure as the data set used to fit the object.
- **type**: [character vector] the type of predicted value. Choices are
  - "hazard" the baseline hazard function when argument newdata is not used and the hazard function when argument newdata is used.
  - "cumhazard" the cumulative baseline hazard function when argument newdata is not used and the cumulative hazard function when argument newdata is used.
  - "survival" the survival baseline hazard function when argument newdata is not used and the cumulative hazard function when argument newdata is used.

Several choices can be combined in a vector of strings that match (no matter the case) strings "hazard","cumhazard","survival".

- **keep.strata**: [logical] If TRUE add the (newdata) strata to the output. Only if there any.
- **keep.infoVar**: [logical] For internal use.
- **...**: additional arguments to be passed to predictCox.

Details

Note: the iid and standard errors are computed using the exponential approximation.
Examples

```r
library(survival)

#### generate data ####
set.seed(10)
d <- sampleData(40, outcome="survival")
nd <- sampleData(4, outcome="survival")
d$time <- round(d$time, 1)

#### Cox model ####
fit <- coxph(Surv(time, event) ~ X1 + X2 + X6,
data=d, ties="breslow", x = TRUE, y = TRUE)

## exponential approximation
predictCox(fit, newdata = d, times = 1:5)

## product limit
predictCoxPL(fit, newdata = d, times = 1:5)

#### stratified Cox model ####
fitS <- coxph(Surv(time, event) ~ X1 + strata(X2) + X6,
data=d, ties="breslow", x = TRUE, y = TRUE)

## exponential approximation
predictCox(fitS, newdata = d, times = 1:5)

## product limit
predictCoxPL(fitS, newdata = d, times = 1:5)

#### fully stratified Cox model ####
fitS <- coxph(Surv(time, event) ~ 1,
data=d, ties="breslow", x = TRUE, y = TRUE)

## product limit
GS <- survfit(Surv(time, event)~1, data = d)
range(predictCoxPL(fitS)$survival - GS$surv)

fitS <- coxph(Surv(time, event) ~ strata(X2),
data=d, ties="breslow", x = TRUE, y = TRUE)

## product limit
GS <- survfit(Surv(time, event)~X2, data = d)
range(predictCoxPL(fitS)$survival - GS$surv)
```
Description

Extract event probabilities from fitted regression models and machine learning objects. The function predictRisk is a generic function, meaning that it invokes specifically designed functions depending on the 'class' of the first argument. See predictRisk.

Usage

predictRisk(object, newdata, ...)  
## Default S3 method:  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'double'  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'integer'  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'factor'  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'numeric'  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'glm'  
predictRisk(object, newdata, iid = FALSE, average.iid = FALSE, ...)  
## S3 method for class 'formula'  
predictRisk(object, newdata, ...)  
## S3 method for class 'BinaryTree'  
predictRisk(object, newdata, ...)  
## S3 method for class 'lrm'  
predictRisk(object, newdata, ...)  
## S3 method for class 'rpart'  
predictRisk(object, newdata, ...)  
## S3 method for class 'randomForest'  
predictRisk(object, newdata, ...)  
## S3 method for class 'matrix'  
predictRisk(object, newdata, times, cause, ...)  
## S3 method for class 'aalen'  
predictRisk(object, newdata, times, ...)
predictRisk

## S3 method for class 'cox.aalen'
predictRisk(object, newdata, times, ...)

## S3 method for class 'coxph'
predictRisk(
  object,
  newdata,
  times,
  product.limit = FALSE,
  iid = FALSE,
  average.iid = FALSE,
  ...
)

## S3 method for class 'coxphTD'
predictRisk(object, newdata, times, landmark, ...)

## S3 method for class 'CSCTD'
predictRisk(object, newdata, times, cause, landmark, ...)

## S3 method for class 'coxph.penal'
predictRisk(object, newdata, times, ...)

## S3 method for class 'cph'
predictRisk(
  object,
  newdata,
  times,
  product.limit = FALSE,
  iid = FALSE,
  average.iid = FALSE,
  ...
)

## S3 method for class 'selectCox'
predictRisk(object, newdata, times, ...)

## S3 method for class 'prodlim'
predictRisk(object, newdata, times, cause, ...)

## S3 method for class 'survfit'
predictRisk(object, newdata, times, ...)

## S3 method for class 'psm'
predictRisk(object, newdata, times, ...)

## S3 method for class 'ranger'
predictRisk(object, newdata, times, cause, ...)
Arguments

object A fitted model from which to extract predicted event probabilities.
newdata A data frame containing predictor variable combinations for which to compute predicted event probabilities.
... Additional arguments that are passed on to the current method.
times A vector of times in the range of the response variable, for which the cumulative incidences event probabilities are computed.
cause Identifies the cause of interest among the competing events.
iid Should the iid decomposition be output using an attribute?
predictRisk

average.iid Should the average iid decomposition be output using an attribute?

product.limit If TRUE the survival is computed using the product limit estimator. Otherwise the exponential approximation is used (i.e. exp(-cumulative hazard)).

landmark The starting time for the computation of the cumulative risk.

Details

In uncensored binary outcome data there is no need to choose a time point.

When operating on models for survival analysis (without competing risks) the function still predicts the risk, as $1 - S(t|X)$ where $S(t|X)$ is survival chance of a subject characterized by $X$.

When there are competing risks (and the data are right censored) one needs to specify both the time horizon for prediction (can be a vector) and the cause of the event. The function then extracts the absolute risks $F_c(t|X)$ aka the cumulative incidence of an event of type/cause $c$ until time $t$ for a subject characterized by $X$. Depending on the model it may or not be possible to predict the risk of all causes in a competing risks setting. For example, a cause-specific Cox (CSC) object allows to predict both cases whereas a Fine-Gray regression model (FGR) is specific to one of the causes.

Value

For binary outcome a vector with predicted risks. For survival outcome with and without competing risks a matrix with as many rows as \texttt{NROW(newdata)} and as many columns as \texttt{length(times)}. Each entry is a probability and in rows the values should be increasing.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```r
## binary outcome
library(rms)
set.seed(7)
d <- sampleData(80,outcome="binary")
nd <- sampleData(80,outcome="binary")
fit <- lrm(Y~X1+X8,data=d)
predictRisk(fit,newdata=nd)
## Not run:
library(SuperLearner)
set.seed(1)
sl = SuperLearner(Y = d$Y, X = d[,-1], family = binomial(),
                   SL.library = c("SL.mean", "SL.glmnet", "SL.randomForest"))

## End(Not run)

## survival outcome
# generate survival data
library(prodlim)
set.seed(100)
d <- sampleData(100,outcome="survival")
d[,X1:=as.numeric(as.character(X1))]
```

d[,X2:=as.numeric(as.character(X2))]
# then fit a Cox model
library(rms)
cphmodel <- cph(Surv(time,event)-X1+X2,data=d,surv=TRUE,x=TRUE,y=TRUE)
# or via survival
library(survival)
coxphmodel <- coxph(Surv(time,event)-X1+X2,data=d,x=TRUE,y=TRUE)

# Extract predicted survival probabilities
# at selected time-points:
# for selected predictor values:
ttt <- quantile(d$time)
ndat <- data.frame(X1=c(0.25,0.25,-0.05,0.05),X2=c(0,1,0,1))
# as follows
predictRisk(cphmodel,newdata=ndat,times=ttt)
predictRisk(coxphmodel,newdata=ndat,times=ttt)

# stratified cox model
sfit <- coxph(Surv(time,event)-strata(X1)+X2,data=d,x=TRUE,y=TRUE)
predictRisk(sfit,newdata=d[1:3,],times=c(1,3,5,10))

## simulate learning and validation data
learndat <- sampleData(100,outcome="survival")
valdat <- sampleData(100,outcome="survival")
## use the learning data to fit a Cox model
library(survival)
fitCox <- coxph(Surv(time,event)-X1+X2,data=learndat,x=TRUE,y=TRUE)
## suppose we want to predict the survival probabilities for all subjects
## in the validation data at the following time points:
## 0, 12, 24, 36, 48, 60
psurv <- predictRisk(fitCox,newdata=valdat,times=seq(0,60,12))
## This is a matrix with event probabilities (1-survival)
## one column for each of the 5 time points
## one row for each validation set individual

# Do the same for a randomSurvivalForest model
# library(randomForestSRC)
# rsfmodel <- rfsrc(Surv(time,event)-X1+X2,data=learndat)
# prsfsurv=predictRisk(rsfmodel,newdata=valdat,times=seq(0,60,12))

# Cox with ridge option
f1 <- coxph(Surv(time,event)-X1+X2,data=learndat,x=TRUE,y=TRUE)
f2 <- coxph(Surv(time,event)-ridge(X1)+ridge(X2),data=learndat,x=TRUE,y=TRUE)
## Not run:
plot(predictRisk(f1,newdata=valdat,times=10),
     riskRegression:::predictRisk.coxph(f2,newdata=valdat,times=10),
     xlim=c(0,1),
     ylim=c(0,1),
     xlab="Unpenalized predicted survival chance at 10",
     ylab="Ridge predicted survival chance at 10")

## End(Not run)
## competing risks

```r
library(survival)
library(riskRegression)
library(prodlim)
train <- prodlim::SimCompRisk(100)
test <- prodlim::SimCompRisk(10)
cox.fit <- CSC(Hist(time,cause)~X1+X2,data=train)
predictRisk(cox.fit,newdata=test,times=seq(1:10),cause=1)

## with strata
cox.fit2 <- CSC(list(Hist(time,cause)~strata(X1)+X2,Hist(time,cause)~X1+X2),data=train)
predictRisk(cox.fit2,newdata=test,times=seq(1:10),cause=1)
```

---

**print.ate**  
*Print Average Treatment Effects*

### Description

Print average treatment effects.

### Usage

```r
## S3 method for class 'ate'
print(x, ...)
```

### Arguments

- `x`  
  object obtained with function `ate`

- `...`  
  passed to summary

### See Also

- `summary.ate` to obtained a more detailed output
- `confint.ate` to compute confidence intervals/bands.
- `ate` to compute the average treatment effects.
print.CauseSpecificCox

*Print of a Cause-Specific Cox regression model*

**Description**

Print of a Cause-Specific Cox regression model

**Usage**

```r
# S3 method for class 'CauseSpecificCox'
print(x, ...)
```

**Arguments**

- `x` Object obtained with CSC
- `...` Passed to `print`

print.FGR

*Print of a Fine-Gray regression model*

**Description**

Print of a Fine-Gray regression model

**Usage**

```r
# S3 method for class 'FGR'
print(x, ...)
```

**Arguments**

- `x` Object fitted with function FGR
- `...` Passed to `cmprsk::summary.crr`
print.influenceTest

Output of the Difference Between Two Estimates

Description
Output of the difference between two estimates.

Usage

## S3 method for class 'influenceTest'
print(x, digits = 3, ...)

Arguments

x            object obtained with the function influenceTest.
digits       [integer, >0] indicating the number of decimal places.
...           Passed to print.

Details
to display confidence intervals/bands, the confint method needs to be applied on the object.

See Also
confint.influenceTest to compute confidence intervals/bands. influenceTest to perform the comparison.

print.IPA

Print IPA object

Description
Print method for IPA

Usage

## S3 method for class 'IPA'
print(x, percent = TRUE, digits = 2, ...)

Arguments

x            Object obtained with IPA
percent      Logical. If TRUE show percentages.
digits       Number of digits
...           passed to print
print.predictCox

Print Predictions From a Cox Model

Description

Print predictions from a Cox model.

Usage

## S3 method for class 'predictCox'
print(x, digits = 3, ...)

Arguments

x        object obtained with the function predictCox.
digits   [integer, >0] indicating the number of decimal places.
...      Passed to print.

Details

to display confidence intervals/bands, the confint method needs to be applied on the object.

See Also

confint.predictCox to compute confidence intervals/bands. predictCox to compute the predicted cumulative hazard/survival.

print.predictCSC

Print Predictions From a Cause-specific Cox Proportional Hazard Regression

Description

Print predictions from a Cause-specific Cox proportional hazard regression.

Usage

## S3 method for class 'predictCSC'
print(x, digits = 3, ...)

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>
Arguments

- **x**: object obtained with the function `predictCox`.
- **digits**: [integer, >0] indicating the number of decimal places.
- **...**: Passed to print.

Details

to display confidence intervals/bands, the `confint` method needs to be applied on the object.

See Also

- `confint.predictCSC` to compute confidence intervals/bands. `predict.CauseSpecificCox` to compute the predicted risks.

---

**print.riskRegression**  
*Print function for riskRegression models*

Description

Print function for riskRegression models

Usage

```r
## S3 method for class 'riskRegression'
print(x, times, digits = 3, eps = 10^-4, verbose = TRUE, conf.int = 0.95, ...)
```

Arguments

- **x**: Object obtained with ARR, LRR or riskRegression
- **times**: Time points at which to show time-dependent coefficients
- **digits**: Number of digits for all numbers but p-values
- **eps**: p-values smaller than this number are shown as such
- **verbose**: Level of verbosity
- **conf.int**: level of confidence. default is 0.95
- **...**: not used
print.score

Print Score object

Description
Print method for risk prediction scores

Usage
## S3 method for class 'Score'
print(x, digits, percent = TRUE, ...)

Arguments
x Object obtained with Score.list
digits Number of digits
percent Logical. If TRUE show percentages.
... passed to print

print.subjectWeights

Print subject weights

Description
Print subject weights

Usage
## S3 method for class 'subjectWeights'
print(x, digits = 3, ...)

Arguments
x Subject weights
digits Digits
... not used
**reconstructData**  
*Reconstruct the original dataset*

**Description**  
Reconstruct the original dataset from the elements stored in the coxph object.

**Usage**  
```r
reconstructData(object)
```

**Arguments**  
- **object** a coxph object.

**Author(s)**  
Brice Ozenne broz@sund.ku.dk and Thomas A. Gerds tag@biostat.ku.dk

---

**riskLevelPlot**  
*Level plots for risk prediction models*

**Description**  
Level plots for predicted risks.

**Usage**  
```r
riskLevelPlot(
  object,  
  formula,  
  data = parent.frame(),  
  horizon = NULL,  
  cause = 1,  
  ...  
)
```

**Arguments**  
- **object** risk prediction model object  
- **formula** formula  
- **data** data  
- **horizon** time point  
- **cause** cause of interest  
- **...** passed to lattice::levelplot
Details

Level plots for predicted risks

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```r
# -------------- logistic regression -----------------
expit <- function(x){exp(x)/(1+exp(x))}
partyData <- function(N){
  Age <- runif(N,.5,15)
  Parasites <- rnorm(N,mean=3.5-0.03*Age)
  Fever <- factor(rbinom(N,1,expit(-3.5-.3*Age+.55*Parasites+0.15*Age*Parasites)))
  data.frame(Fever,Age,Parasites)
}
d <- partyData(100)
f <- glm(Fever~Age+Parasites,data=d,family="binomial")
riskLevelPlot(f,Fever~Age+Parasites,d)
library(randomForest)
rf <- randomForest(Fever~Age+Parasites,data=d)
riskLevelPlot(f,Fever~Age+Parasites,d)
riskLevelPlot(rf,Fever~Age+Parasites,d)

# -------------- survival analysis -----------------
# --simulate an artificial data frame
# with survival response and three predictors

library(survival)
library(prodlim)
set.seed(140515)
sdat <- sampleData(43,outcome="survival")
# -- fit a Cox regression model
survForm = Surv(time,event) ~ X8 + X9
cox <- coxph(survForm, data = sdat,x=TRUE)
# --choose a time horizon for the predictions and plot the risks
timeHorizon <- floor(median(sdat$time))
riskLevelPlot(cox, survForm, data = sdat, horizon = timeHorizon)

# -------------- competing risks -----------------
# -- simulate an artificial data frame
# with competing cause response and three predictors

library(cmprsk)
library(riskRegression)
set.seed(140515)
crdat <- sampleData(49)
```
riskRegression

Risk Regression Fits a regression model for the risk of an event – allowing for competing risks.

Description

This is a wrapper for the function `comp.risk` from the timereg package. The main difference is one marks variables in the formula that should have a time-dependent effect whereas in `comp.risk` one marks variables that should have a time constant (proportional) effect.

Usage

```r
riskRegression(
  formula,
  data,
  times,
  link = "relative",
  cause,
  conf.int = TRUE,
  cens.model,
  cens.formula,
  max.iter = 50,
  conservative = TRUE,
  ...)
```

Arguments

- `formula`: Formula where the left hand side specifies the event history `event.history` and the right hand side the linear predictor. See examples.
- `data`: The data for fitting the model in which includes all the variables included in formula.
- `times`: Vector of times. For each time point in `times` estimate the baseline risk and the timevarying coefficients.
- `cause`: The cause of interest.
conf.int If TRUE return the iid decomposition, that can be used to construct confidence bands for predictions.

cens.model Specified the model for the (conditional) censoring distribution used for deriving weights (IFPW). Defaults to "KM" (the Kaplan-Meier method ignoring covariates) alternatively it may be "Cox" (Cox regression).

cens.formula Right hand side of the formula used for fitting the censoring model. If not specified the right hand side of formula is used.

max.iter Maximal number of iterations.

conservative If TRUE use variance formula that ignores the contribution by the estimate of the inverse of the probability of censoring weights

... Further arguments passed to comp.risk

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>, Thomas H. Scheike <ts@biostat.ku.dk>

References


Scheike, Zhang and Gerds (2008), Predicting cumulative incidence probability by direct binomial regression, Biometrika, 95, 205-220.


Examples

library(prodlim)
data(Melanoma,package="riskRegression")
## tumor thickness on the log-scale
Melanoma$logthick <- log(Melanoma$thick)

# Single binary factor

## absolute risk regression
library(survival)
library(prodlim)
fit.arr <- ARR(Hist(time,status)-sex,data=Melanoma,cause=1)
print(fit.arr)
  # show predicted cumulative incidences
plot(fit.arr,col=3:4,newdata=data.frame(sex=c("Female","Male")))

## compare with non-parametric Aalen-Johansen estimate
library(prodlim)
fit.aj <- prodlim(Hist(time,status)-sex,data=Melanoma)
plot(fit.aj,conf.int=FALSE)
plot(fit.arr, add=TRUE, col=3:4, newdata=data.frame(sex=c("Female","Male")))

## with time-dependent effect
fit.tarr <- ARR(Hist(time,status)~strata(sex), data=Melanoma, cause=1)
plot(fit.tarr, newdata=data.frame(sex=c("Female","Male")))

## logistic risk regression
fit.lrr <- LRR(Hist(time,status)~sex, data=Melanoma, cause=1)
summary(fit.lrr)

# Single continuous factor

## tumor thickness on the log-scale
Melanoma$logthick <- log(Melanoma$thick)

## absolute risk regression
fit2.arr <- ARR(Hist(time,status)~logthick, data=Melanoma, cause=1)
print(fit2.arr)
# show predicted cumulative incidences
plot(fit2.arr, col=1:5, newdata=data.frame(logthick=quantile(Melanoma$logthick)))

## comparison with nearest neighbor non-parametric Aalen-Johansen estimate
library(prodlim)
fit2.aj <- prodlim(Hist(time,status)~logthick, data=Melanoma)
plot(fit2.aj, conf.int=FALSE, newdata=data.frame(logthick=quantile(Melanoma$logthick)))
plot(fit2.arr, add=TRUE, col=1:5, lty=3, newdata=data.frame(logthick=quantile(Melanoma$logthick)))

## logistic risk regression
fit2.lrr <- LRR(Hist(time,status)~logthick, data=Melanoma, cause=1)
summary(fit2.lrr)

## change model for censoring weights
library(rms)
fit2a.lrr <- LRR(Hist(time,status)~logthick, data=Melanoma, cause=1, cens.model="cox", cens.formula=~sex+epicel+ulcer+age+logthick)
summary(fit2a.lrr)

## compare prediction performance
Score(list(ARR=fit2.arr, AJ=fit2.aj, LRR=fit2.lrr), formula=Hist(time,status)~1, data=Melanoma)

# multiple regression
library(riskRegression)
library(prodlim)
# absolute risk model
multi.arr <- ARR(Hist(time,status)~logthick+sex+age+ulcer, data=Melanoma, cause=1)

# stratified model allowing different baseline risk for the two gender
multi.arr <- ARR(Hist(time,status)~thick+strata(sex)+age+ulcer, data=Melanoma, cause=1)
# stratify by a continuous variable: strata(age)
multi.arr <- ARR(Hist(time,status)~tp(thick,power=0)+strata(age)+sex+ulcer,
    data=Melanoma,
    cause=1)

fit.arr2a <- ARR(Hist(time,status)~tp(thick,power=1),data=Melanoma,cause=1)
summary(fit.arr2a)
fit.arr2b <- ARR(Hist(time,status)~timevar(thick),data=Melanoma,cause=1)
summary(fit.arr2b)

## logistic risk model
fit.lrr <- LRR(Hist(time,status)~thick,data=Melanoma,cause=1)
summary(fit.lrr)

## nearest neighbor non-parametric Aalen-Johansen estimate
library(prodlim)
fit.aj <- prodlim(Hist(time,status)~thick,data=Melanoma)
plot(fit.aj,conf.int=FALSE)

# prediction performance
x <- Score(list(fit.arr2a,fit.arr2b,fit.lrr),
    data=Melanoma,
    formula=Hist(time,status)-1,
    cause=1,
    split.method="none")

---

**riskRegression.options**

*Global options for riskRegression*

**Description**

Output and set global options for the riskRegression package.

**Usage**

`riskRegression.options(...)`

**Arguments**

... for now limited to method.predictRisk and method.predictRiskIID.
Details

only used by the ate function.

Examples

options <- riskRegression.options()

## add new method.predictRiskIID
riskRegression.options(method.predictRiskIID = c(options$method.predictRiskIID,"xx"))

riskRegression.options()

---

description

Fast computation of sweep(X, MARGIN = 2, FUN = "+", STATS = center)

Usage

rowCenter_cpp(X, center)

Arguments

X  A matrix.

center a numeric vector of length equal to the number of rows of x

Value

A matrix of same size as X.

Author(s)

Brice Ozenne <broz@sund.ku.dk>

Examples

x <- matrix(1,6,5)
sweep(x, MARGIN = 2, FUN = "+", STATS = 1:5)
rowCenter_cpp(x, 1:5 )

rowCenter_cpp(x, colMeans(x) )
rowCumProd

**Description**
Fast computation of `t(apply(x, 1, cumprod))`

**Usage**
```
rowCumProd(x)
```

**Arguments**
- `x`
  A matrix.

**Value**
A matrix of same size as `x`.

**Author(s)**
Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**
```
x <- matrix(1:8, ncol=2)
rowCumProd(x)
```

---

rowCumSum

**Apply cumsum in each row**

**Description**
Fast computation of `t(apply(x, 1, cumsum))`

**Usage**
```
rowCumSum(x)
```

**Arguments**
- `x`
  A matrix.

**Value**
A matrix of same size as `x`. 
**rowMultiply_cpp**

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```r
x <- matrix(1:8, ncol=2)
rowCumSum(x)
```

---

**Description**

Fast computation of `sweep(X, MARGIN = 2, FUN = "*", STATS = scale)`

**Usage**

```r
rowMultiply_cpp(X, scale)
```

**Arguments**

- `X` A matrix.
- `scale` a numeric vector of length equal to the number of rows of `x`

**Value**

A matrix of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```r
x <- matrix(1, 6, 5)
sweep(x, MARGIN = 2, FUN = "*", STATS = 1:5)
rowMultiply_cpp(x, 1:5)
rowMultiply_cpp(x, 1/colMeans(x))
```
### rowPaste

**Collapse Rows of Characters.**

**Description**

Collapse rows of characters. Fast alternative to `apply(x,1,paste0,collapse="")`

**Usage**

```r
rowPaste(object)
```

**Arguments**

- `object` A matrix/data.frame/list containing the characters.

**Examples**

```r
## Not run:
M <- matrix(letters,nrow = 26, ncol = 2)
rowPaste(M)
## End(Not run)
```

### rowScale_cpp

**Apply / by row**

**Description**

Fast computation of `sweep(X, MARGIN = 2, FUN = "/", STATS = scale)`

**Usage**

```r
rowScale_cpp(X, scale)
```

**Arguments**

- `X` A matrix.
- `scale` A numeric vector of length equal to the number of rows of `X`.

**Value**

A matrix of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>
Examples

```r
x <- matrix(1, 6, 5)
sweep(x, MARGIN = 2, FUN = "/", STATS = 1:5)
rowScale_cpp(x, 1:5)

rowScale_cpp(x, colMeans(x))
```

---

**rowSumsCrossprod**

*Apply crossprod and rowSums*

Description

Fast computation of crossprod(rowSums(X), Y)

Usage

```r
rowSumsCrossprod(X, Y, transposeY)
```

Arguments

- `X` A matrix with dimensions n\*k. Hence the result of `rowSums(X)` has length n.
- `Y` A matrix with dimensions n\*m. Can be a matrix with dimension m\*n but then `transposeY` should be TRUE.
- `transposeY` Logical. If TRUE transpose Y before matrix multiplication.

Value

A vector of length m.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```r
x <- matrix(1:10, nrow=5)
y <- matrix(1:20, ncol=4)
rowSumsCrossprod(x, y, 0)
```

```r
x <- matrix(1:10, nrow=5)
y <- matrix(1:20, ncol=5)
rowSumsCrossprod(x, y, 1)
```
sampleData

Simulate data with binary or time-to-event outcome

Description

Simulate data with binary outcome and 10 covariates.

Usage

```
sampleData(n, outcome="competing.risks",
formula= ~ f(X1,2)+f(X2,-0.033)+f(X3,0.4)+f(X6,.1)+f(X7,-.1)+f(X8,.5)+f(X9,-1))
sampleDataTD(n, n.intervals=5, outcome="competing.risks",
formula= ~ f(X1,2)+f(X2,-0.033)+f(X3,0.4)+f(X6,.1)+f(X7,-.1)+f(X8,.5)+f(X9,-1))
```

Arguments

- `n`: Sample size
- `outcome`: Character vector. Response variables are generated according to keywords: "binary" = binary response, "survival" = survival response, "competing.risks" = competing risks response
- `formula`: Specify regression coefficients
- `n.intervals`: sampleDataTD only: the maximum number of episodes in which the covariates are updated.

Details

For the actual lava::regression parameters see the function definition.

Value

Simulated data as data.table with n rows and the following columns: Y (binary outcome), time (non-binary outcome), event (non-binary outcome), X1-X5 (binary predictors), X6-X10 (continuous predictors)

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

lvm

Examples

```
sampleData(10, outcome="binary")
sampleData(10, outcome="survival")
sampleData(10, outcome="competing.risks")
```
Description

Methods to score the predictive performance of risk markers and risk prediction models

Usage

```r
## S3 method for class 'list'
Score(
  object,
  formula,
  data,
  metrics = c("auc", "brier"),
  summary = NULL,
  plots = NULL,
  cause,
  times,
  landmarks,
  use.event.times = FALSE,
  null.model = TRUE,
  se.fit = TRUE,
  conservative = FALSE,
  multi.split.test = FALSE,
  conf.int = 0.95,
  contrasts = TRUE,
  probs = c(0, 0.25, 0.5, 0.75, 1),
  cens.method = "ipcw",
  cens.model = "cox",
  split.method,
  B,
  M,
  seed,
  trainseeds,
  parallel = c("no", "multicore", "snow"),
  ncpus = 1,
  cl = NULL,
  progress.bar = 3,
  keep,
  predictRisk.args,
  debug = 0L,
  useEventTimes,
  nullModel,
  censMethod,
  censModel,
  splitMethod,
)
Arguments

object

List of risk predictions (see details and examples).

formula

A formula which identifies the outcome (left hand side). E.g., \(Y \sim 1\) for binary and \(\text{Hist}(\text{time}, \text{status}) \sim 1\) for time-to-event outcome. In right censored data, the right hand side of the formula is used to estimate the inverse probability of censoring weights (IPCW) model.

data

data.frame or data.table in which the formula can be interpreted.

metrics

Character vector specifying which metrics to apply. Case does not matter. Choices are "AUC" and "Brier".

summary

Character vector specifying which summary statistics to apply to the predicted risks. Choices are "risks", "IPA", "riskQuantile" and "ibs". Can be all c("risks","IPA","riskQuantile","ibs") or a subset thereof.

- "risks" adds the predicted risks to the output.
- "ipa" computes the index of prediction accuracy (AKA R-squared) based on Brier scores for model vs null model
- "riskQuantile" calculates time-point specific boxplots for the predicted risks (or biomarker values) conditional on the outcome at the time-point.
- "ibs" calculates integrated Brier scores across the time points at which the Brier score is computed. This works only with time-to-event outcome and the results depend on the argument times.

Set to NULL to avoid estimation of summary statistics.

plots

Character vector specifying for which plots to put data into the result. Currently implemented are "ROC", "Calibration" and "boxplot". In addition, one can plot AUC and Brier score as function of time as soon as times has at least two different values.

cause

Event of interest. Used for binary outcome \(Y\) to specify that risks are risks of the event \(Y=\text{event}\) and for competing risks outcome to specify the cause of interest.

times

For survival and competing risks outcome: list of prediction horizons. All times which are greater than the maximal observed time in the data set are automatically removed. Note that the object returned by the function may become huge when the prediction performance is estimated at many prediction horizons.

landmarks

Not yet implemented.

use.event.times

If TRUE merge all unique event times with the vector given by argument times.

null.model

If TRUE fit a risk prediction model which ignores the covariates and predicts the same value for all subjects. The model is fitted using data and the left hand side of formula. For binary outcome this is just the empirical prevalence. For (right censored) time to event outcome, the null models are equal to the Kaplan-Meier estimator (no competing risks) and the Aalen-Johansen estimator (with competing risks).
se.fit Logical or 0 or 1. If FALSE or 0 do not calculate standard errors.

conservative Logical, only relevant in right censored data. If TRUE ignore variability of the estimate of the inverse probability of censoring weights when calculating standard errors for prediction performance parameters. This can potentially reduce computation time and memory usage at a usually very small expense of a slightly higher standard error.

multi.split.test Logical or 0 or 1. If FALSE or 0 do not calculate multi-split tests. This argument is ignored when split.method is "none".

conf.int Either logical or a numeric value between 0 and 1. In right censored data, confidence intervals are based on Blanche et al (see references). Setting FALSE prevents the computation confidence intervals. TRUE means compute 95 percent confidence intervals and corresponding p-values for AUC and Brier score. If set to 0.87, the level of significance is 13 percent. So, do not set it to 0.87.

contrasts Either logical or a list of contrasts. A list of contrasts defines which risk prediction models (markers) should be contrasted with respect to their prediction performance. If TRUE do all possible comparisons. For example, when object is a list with two risk prediction models and null.model=TRUE setting TRUE is equivalent to list(c(0,1,2),c(1,2)) where c(0,1,2) codes for the two comparisons: 1 vs 0 and 2 vs 0 (positive integers refer to elements of object, 0 refers to the benchmark null model which ignores the covariates). This again is equivalent to explicitly setting list(c(0,1),c(0,2),c(1,2)). A more complex example: Suppose object has 7 elements and you want to do the following 3 comparisons: 6 vs 3, 2 vs 5 and 2 vs 3, you should set contrasts=c(6,3),c(2,5,3).

probs Quantiles for retrospective summary statistics of the predicted risks. This affects the result of the function boxplot.Score.

cens.method Method for dealing with right censored data. Either "ipcw" or "pseudo". Here IPCW refers to inverse probability of censoring weights and pseudo for jackknife pseudo values. Right now pseudo values are only used for calibration curves.

cens.model Model for estimating inverse probability of censored weights. Implemented are the Kaplan-Meier method ("km") and Cox regression ("cox") both applied to the censored times. If the right hand side of formula does not specify covariates, the Kaplan-Meier method is used even if this argument is set to "cox".

split.method Method for cross-validation. Right now the only choice is bootcv in which case bootstrap learning sets are drawn with our without replacement (argument M) from data. The data not included in the current bootstrap learning set are used as validation set to compute the prediction performance.

B Number of bootstrap sets for cross-validation.

M Size of subsamples for bootstrap cross-validation. If specified it has to be an integer smaller than the size of data.

seed Super seed for setting training data seeds when randomly splitting (bootstrapping) the data during cross-validation.

trainseeds Seeds for training models during cross-validation.

parallel The type of parallel operation to be used (if any). If missing, the default is "no".
ncpus
integer: number of processes to be used in parallel operation.

c1
An optional parallel or snow cluster for use if parallel = "snow". If not supplied, a cluster on the local machine is created for the duration of the Score call.

progress.bar
Style for txtProgressBar. Can be 1, 2, 3 see help(txtProgressBar) or NULL to avoid the progress bar.

keep
list of characters (not case sensitive) which determines additional output. "residuals" provides Brier score residuals and "splitindex" provides sampling index used to split the data into training and validation sets. "vcov" provides the variance-covariance matrix of the estimated parameters.

predictRisk.args
A list of argument-lists to control how risks are predicted. The names of the lists should be the S3-classes of the object. The argument-lists are then passed on to the S3-class specific predictRisk method. For example, if your object contains one or several random forest model fitted with the function randomForestSRC::rfsrc then you can specify additional arguments for the function riskRegression::predictRisk::rfsrc which will pass these on to the function randomForestSRC::predict.rfsrc. A specific example in this case would be list(rfsrc=list(na.action="na.impute").

A more flexible approach is to write a new predictRisk S3-method. See Details.

debug
Logical. If TRUE indicate landmark in progress of the program.

useEventTimes obsolete.

nullModel obsolete.

censMethod obsolete.

censModel obsolete.

splitMethod obsolete.

... Named list containing additional arguments that are passed on to the predictRisk methods corresponding to object. See examples.

Details

The function implements a toolbox for the risk prediction modeller: all tools work for the three outcomes: (1) binary (uncensored), (2) right censored time to event without competing risks, (3) right censored time to event with competing risks.

Computed are the (time-dependent) Brier score and the (time-dependent) area under the ROC curve for a list of risk prediction models either in external validation data or in the learning data using bootstrap cross-validation. The function optionally provides results for plotting (time-point specific) ROC curves, for (time-point specific) calibration curves and for (time-point specific) retrospective boxplots.

For uncensored binary outcome the Delong-Delong test is used to contrast AUC of rival models. In right censored survival data (with and without competing risks) the p-values correspond to Wald tests based on standard errors obtained with an estimate of the influence function as described in detail in the appendix of Blanche et al. (2015).

This function works with one or multiple models that predict the risk of an event R(t|X) for a subject characterized by predictors X at time t. With binary endpoints (outcome 0/1 without time
component) the risk is simply R(X). In case of a survival object without competing risks the function still works with predicted event probabilities, i.e., R(t|X)=1-S(t|X) where S(t|X) is the predicted survival chance for subject X at time t.

The already existing predictRisk methods (see methods(predictRisk)) may not cover all models and methods for predicting risks. But users can quickly extend the package as explained in detail in Mogensen et al. (2012) for the predecessors pec::predictSurvProb and pec::predictEventProb which have been unified as riskRegression::predictRisk.

Bootstrap Crossvalidation (see also Gerds & Schumacher 2007 and Mogensen et al. 2012)

B=10, M (not specified or M=NROW(data)) Training of each of the models in each of 10 bootstrap data sets (learning data sets). Learning data sets are obtained by sampling NROW(data) subjects of the data set with replacement. There are roughly .632*NROW(data) subjects in the learning data (inbag) and .368*NROW(data) subjects not in the validation data sets (out-of-bag).

These are used to estimate the scores: AUC, Brier, etc. Reported are averages across the 10 splits.

```r
## Bootstrap with replacement
set.seed(13) N=17 data = data.frame(id=1:N,y=rbinom(N,1,.3),x=rnorm(N))
boot.index = sample(1:N,size=N,replace=TRUE) boot.index inbag = 1:N outofbag = !inbag
learn.data = data[inbag] val.data = data[outofbag] riskRegression:::getSplitMethod("bootcv",B=10,N=17)$index
```

Value

List with scores and assessments of contrasts, i.e., tests and confidence limits for performance and difference in performance (AUC and Brier), summaries and plots. Most elements are in data.table format.

Author(s)

Thomas A Gerds <tag@biostat.ku.dk> and Paul Blanche <paul.blanche@univ-ubs.fr>

References


Paul Blanche, Cecile Proust-Lima, Lucie Loubere, Claudine Berr, Jean-Francois Dartigues, and Helene Jacqmin-Gadda. Quantifying and comparing dynamic predictive accuracy of joint models


E. Graf et al. (1999), Assessment and comparison of prognostic classification schemes for survival data. Statistics in Medicine, vol 18, pp= 2529–2545.


Examples

# binary outcome
library(lava)
set.seed(18)
learndat <- sampleData(48,outcome="binary")
testdat <- sampleData(40,outcome="binary")

## score logistic regression models
lr1 = glm(Y~X1+X2+X7+X9,data=learndat,family=binomial)
lr2 = glm(Y~X3+X5,data=learndat,family=binomial)
Score(list("LR(X1+X2+X7+X9)"=lr1,"LR(X3+X5)"=lr2),formula=Y~1,data=testdat)

## ROC curve and calibration plot
xb=Score(list("LR(X1+X2+X7+X9)"="lr1","LR(X3+X5+X6)"="lr2"),formula=Y~1, data=testdat,plots=c("calibration","ROC"))
## Not run: plotROC(xb)
plotCalibration(xb)

## End(Not run)

## compute AUC for a list of continuous markers
markers = as.list(testdat[,.(X6,X7,X8,X9,X10)])
Score(markers,formula=Y~1,data=testdat,metrics=c("auc"))

# cross-validation
## Not run:
learndat=sampleData(400,outcome="binary")
lr1a = glm(Y~X6,data=learndat,family=binomial)
lr2a = glm(Y~X7+X8+X9,data=learndat,family=binomial)

# bootstrap cross-validation
x1=Score(list("LR1"=lr1a,"LR2"=lr2a),formula=Y~1,data=learndat,split.method="bootcv",B=100)
x1

# leave-one-out and leave-pair-out bootstrap
x2=Score(list("LR1"=lr1a,"LR2"=lr2a),formula=Y~1,data=learndat,split.method="loob",B=100,plots="calibration")

## End(Not run)

# survival outcome

# Score Cox regression models
## Not run: library(survival)
library(rms)
library(prodlim)
set.seed(18)
trainSurv <- sampleData(100,outcome="survival")
testSurv <- sampleData(40,outcome="survival")
cox1 = coxph(Surv(time,event)~X1+X2+X7+X9,data=trainSurv, y=TRUE, x = TRUE)
cox2 = coxph(Surv(time,event)~X3+X5+X6,data=trainSurv, y=TRUE, x = TRUE)
xs=Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
formula=Surv(time,event)-1,data=testSurv,conf.int=FALSE,times=c(5,8))
xs

## End(Not run)

# Integrated Brier score
## Not run:
x=Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
  formula=Surv(time,event)-1,data=testSurv,conf.int=FALSE,
  summary="ibs",times=sort(unique(testSurv$time)))

## End(Not run)

# time-dependent AUC for list of markers
## Not run:
survmarkers = as.list(testSurv[,.(X6,X7,X8,X9,X10)])
Score(survmarkers,
  formula=Surv(time,event)-1,metrics="auc",data=testSurv,
  conf.int=TRUE,times=c(5,8))

# compare models on test data
Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
  formula=Surv(time,event)-1,data=testSurv,conf.int=TRUE,times=c(5,8))

## End(Not run)

# crossvalidation models in traindata
## Not run:
library(survival)
set.seed(18)
score.list

```r
# Cox regression
trainSurv <- sampleData(400, outcome="survival")
cox1 = coxph(Surv(time,event) ~ X1+X2+X7+X9, data=trainSurv, y=TRUE, x = TRUE)
cox2 = coxph(Surv(time,event) ~ X3+X5+X6, data=trainSurv, y=TRUE, x = TRUE)
x1 = Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
         formula=Surv(time,event) ~ 1, data=trainSurv, conf.int=TRUE, times=c(5,8),
         split.method="loob", B=100, plots="calibration")

x2 = Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
         formula=Surv(time,event) ~ 1, data=trainSurv, conf.int=TRUE, times=c(5,8),
         split.method="bootcv", B=100)

# Competing risks outcome
set.seed(18)
trainCR <- sampleData(40, outcome="competing.risks")
testCR <- sampleData(40, outcome="competing.risks")
library(riskRegression)
library(cmprsk)
# Cause-specific Cox regression
csc1 = CSC(Hist(time,event) ~ X1+X2+X7+X9, data=trainCR)
csc2 = CSC(Hist(time,event) ~ X3+X5+X6, data=trainCR)
# Fine-Gray regression
fgr1 = FGR(Hist(time,event) ~ X1+X2+X7+X9, data=trainCR, cause=1)
fgr2 = FGR(Hist(time,event) ~ X3+X5+X6, data=trainCR, cause=1)
Score(list("CSC(X1+X2+X7+X9)"=csc1,"CSC(X3+X5+X6)"=csc2,
         "FGR(X1+X2+X7+X9)"=fgr1,"FGR(X3+X5+X6)"=fgr2),
         formula=Hist(time,event) ~ 1, data=testCR, se.fit=1L, times=c(5,8))
```

```
# Not run:
# reproduce some results of Table IV of Blanche et al. Stat Med 2013
data(Paquid)
ResPaquid <- Score(list("DSST"=-Paquid$DSST,"MMSE"=-Paquid$MMSE),
                     formula=Hist(time,status) ~ 1,
                     data=Paquid,
                     null.model = FALSE,
                     conf.int=TRUE,
                     metrics=c("auc"),
                     times=c(3,5,10),
                     plots="ROC")
ResPaquid
plotROC(ResPaquid, time=5)
```
selectCox

Backward variable selection in the Cox regression model

Description

This is a wrapper function which first selects variables in the Cox regression model using `fastbw` from the `rms` package and then returns a fitted Cox regression model with the selected variables.

Usage

```r
selectCox(formula, data, rule = "aic")
```

Arguments

- **formula**: A formula object with a `Surv` object on the left-hand side and all the variables on the right-hand side.
- **data**: Name of an data frame containing all needed variables.
- **rule**: The method for selecting variables. See `fastbw` for details.

Details

This function first calls `cph` then `fastbw` and finally `cph` again.

References


Examples

```r
library(pec)
library(prodlim)
data(GBSG2)
library(survival)
f <- selectCox(Surv(time,cens)~horTh+age+menostat+tsize+tgrade+pnodes+progrec+estrec ,
               data=GBSG2)
```
selectJump  
*Evaluate the influence function at selected times*

**Description**
Evaluate the influence function at selected times

**Usage**
```
selectJump(IF, times, type)
```

**Arguments**
- **IF**: influence function returned by iidCox
- **times**: the times at which the influence function should be assessed
- **type**: can be "hazard" or/and "cumhazard".

**Value**
An object with the same dimensions as IF

**Author(s)**
Brice Ozenne broz@sund.ku.dk

---

**simActiveSurveillance**  
*Simulate data of a hypothetical active surveillance prostate cancer study*

**Description**
Simulate data of a hypothetical active surveillance prostate cancer study

**Usage**
```
simActiveSurveillance(n)
```

**Arguments**
- **n**: sample size

**Details**
This is based on the functionality of `library(lava)`. 
**Value**

data table of size \( n \)

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**Examples**

```r
set.seed(71)
simActiveSurveillance(3)
```

---

**Description**

Simulate data alike the Melanoma data

**Usage**

```r
simMelanoma(n)
```

**Arguments**

- \( n \)  
  sample size

**Details**

This is based on the functionality of `library(lava)`.

**Value**

data table of size \( n \)

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**Examples**

```r
set.seed(71)
simMelanoma(3)
```
sliceMultiply_cpp  Apply * by slice

Description
Fast computation of sweep(X, MARGIN = 1:2, FUN = "+", STATS = scale)

Usage
sliceMultiply_cpp(X, M)
sliceMultiplyPointer_cpp(X, M)

Arguments
X  An array.
M  A matrix with the same number of row and columns as X.

Value
An array of same size as X.

Author(s)
Brice Ozenne <broz@sund.ku.dk>

Examples
x <- array(1, dim = c(2,6,5))
M <- matrix(1:12,2,6)
sweep(x, MARGIN = 1:2, FUN = "*", STATS = M)
sliceMultiply_cpp(x, M)

sliceScale_cpp  Apply / by slice

Description
Fast computation of sweep(X, MARGIN = 1:2, FUN = "/", STATS = scale)

Usage
sliceScale_cpp(X, M)
sliceScalePointer_cpp(X, M)
Arguments

X    An array.
M    A matrix with the same number of row and columns as X.

Value

An array of same size as X.

Author(s)

Brice Ozenne <broz@sund.ku.dk>

Examples

```r
x <- array(1, dim = c(2,6,5))
M <- matrix(1:12,2,6)
sweep(x, MARGIN = 1:2, FUN = "/", STATS = M)
sliceScale_cpp(x, M)
```

Description

TODO

Usage

SmcFcs(formula, data, m = 5, method, fitter = "glm", fit.formula, ...)

Arguments

formula    TODO
data        TODO
m           TODO
method      TODO
fitter      TODO
fit.formula TODO
...         TODO
# @export

```r
SmcFcs
```
splitStrataVar  

*Reconstruct each of the strata variables*

**Description**
Reconstruct each of the strata variables from the strata variable stored in the coxph object.

**Usage**
```
splitStrataVar(object)
```

**Arguments**
- `object` a coxph object.

**Author(s)**
Brice Ozenne broz@sund.ku.dk and Thomas A. Gerds tag@biostat.ku.dk

---

subjectWeights  

*Estimation of censoring probabilities at subject specific times*

**Description**
This function is used internally to construct pseudo values by inverse of the probability of censoring weights.

**Usage**
```
subjectWeights(
  formula,
  data,
  method = c("cox", "marginal", "km", "nonpar", "forest", "none"),
  args,
  lag = 1
)
```

**Arguments**
- `formula` A survival formula like, Surv(time,status)~1 or Hist(time,status)~1 where status=0 means censored. The status variable is internally reversed for estimation of censoring rather than survival probabilities. Some of the available models, see argument `model`, will use predictors on the right hand side of the formula.
- `data` The data used for fitting the censoring model
- `method` Censoring model used for estimation of the (conditional) censoring distribution.
subjectWeights

args  Arguments passed to the fitter of the method.
lag  If equal to 1 then obtain \( G(T_i \mid X_i) \), if equal to 0 estimate the conditional censoring distribution at the subject.times, i.e. \( G(T_i \mid X_i) \).

Details

Inverse of the probability of censoring weights usually refer to the probabilities of not being censored at certain time points. These probabilities are also the values of the conditional survival function of the censoring time given covariates. The function subjectWeights estimates the conditional survival function of the censoring times and derives the weights.

IMPORTANT: the data set should be ordered, order(time,-status) in order to get the weights in the right order for some choices of method.

Value

- times: The times at which weights are estimated
- weights: Estimated weights at individual time values subject.times
- lag: The time lag.
- fit: The fitted censoring model
- method: The method for modelling the censoring distribution
- call: The call

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```r
library(prodlim)
library(survival)
dat=SimSurv(300)

dat <- dat[order(dat$time,-dat$status),]
# using the marginal Kaplan-Meier for the censoring times
WKMM=subjectWeights(Hist(time,status)~X2,data=dat,method="marginal")
plot(WKMM$fit)
WKMM$fit
WKMM$weights

# using the Cox model for the censoring times given X2
WCox=subjectWeights(Surv(time,status)~X2,data=dat,method="cox")
WCox
plot(WCox$weights,WKMM$weights)

# using the stratified Kaplan-Meier for the censoring times given X2
```
subsetIndex <- subjectWeights(Surv(time,status)-X2,data=dat,method="nonpar")
plot(subsetIndex$fit,add=FALSE)

subsetIndex

Extract Specific Elements From An Object

Description

Extract specific elements from an object.

Usage

subsetIndex(object, index, default, ...)

## Default S3 method:
subsetIndex(object, index, default, ...)

## S3 method for class 'matrix'
subsetIndex(object, index, default, col = TRUE, ...)

Arguments

object A vector or a matrix.
index index of the elements to be extracted. 0 indicates that the column should be set to the default value. NA indicates that the column should be set to NA.
default the default value.
... Only used by the generic method.
col If object is a matrix, TRUE lead to extract the columns and FALSE the rows.

Examples

M <- matrix(rnorm(50),5,10)
subsetIndex(M, index = c(0,0,1), default = 0)
subsetIndex(M, index = c(0,2,3,NA), default = 0)
subsetIndex(M, index = c(0,NA,2,3,NA), default = 0)

C <- 1:10
subsetIndex(C, index = c(0,0,1,5,NA), default = 0)
**summary.ate**

Summary Average Treatment Effects

**Description**

Summary average treatment effects.

**Usage**

```r
## S3 method for class 'ate'
summary(
  object,
  digits = 3,
  type = c("meanRisk", "diffRisk", "ratioRisk"),
  estimator = object$estimator[1],
  ...
)
```

**Arguments**

- `object`: object obtained with function `ate`
- `digits`: [integer, >0] Number of digits.
- `type`: [character vector] what to displayed. Can be any combination of "meanRisk", "diffRisk", and "ratioRisk".
- `estimator`: [character] The type of estimator relative to which the estimates should be displayed.
- `...`: passed to print

**Details**

To display confidence intervals/bands and p.value, the `confint` method needs to be applied on the object.

**See Also**

`confint.ate` to compute confidence intervals/bands. `ate` to compute the average treatment effects.
summary.FGR

Summary of a Fine-Gray regression model

Description
Summary of a Fine-Gray regression model

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

Arguments
object Object fitted with function FGR
... passed to cmprsk::summary.crr

summary.riskRegression

Summary of a risk regression model

Description
Summary of a risk regression model

Usage
## S3 method for class 'riskRegression'
summary(
  object,
  times,
  digits = 3,
  pvalue.digits = 4,
  eps = 10^-4,
  verbose = TRUE,
  ...
)

Arguments
object Object obtained with ARR, LRR or riskRegression
times Time points at which to show time-dependent coefficients
digits Number of digits for all numbers but p-values
pvalue.digits Number of digits for p-values
summary.Score

eps p-values smaller than this number are shown as such
verbose Level of verbosity
... not used

---

Summary of prediction performance metrics

Description
Summarizing a Score object

Usage
## S3 method for class 'Score'
summary(
  object,
  times,
  what = c("score", "contrasts"),
  digits = 1,
  pvalue.digits = 4,
  ...
)

Arguments
  object Object obtained with Score.
  times Select time points
  what Either "score", "contrasts" or both, i.e., c("score", "contrasts")
  digits For rounding everything but p-values
  pvalue.digits For rounding p-values
  ... not used

Details
The AUC and the Brier score are put into tables

Value
List of tables

Author(s)
Thomas A. Gerds <tag@biostat.ku.dk>

See Also
Score
SuperPredictor

*Formula interface for SuperLearner::SuperLearner*

Description

Formula interface for SuperLearner::SuperLearner

Usage

SuperPredictor(
  formula,
  data,
  family = "binomial",
  SL.library = c("SL.glm", "SL.glm.interaction", "SL.ranger"),
  ...
)

Arguments

- **formula**
  where the left hand side specifies the outcome and the right hand side the predictors
- **data**
  data set in which formula can be evaluated
- **family**
  the outcome family. default is binomial
- **SL.library**
  the SuperLearner libraries
- **...**
  passed to SuperLearner::SuperLearner

Details

Formula interface for SuperLearner::SuperLearner

Examples

```r
# Not run:
library(SuperLearner)
library(data.table)
d = sampleData(338, outcome="binary")
spfit = SuperPredictor(Y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10, data=d)
predictRisk(spfit)
x <- Score(list(spfit), data=d, formula=Y~1)

# End(Not run)
```
SurvResponseVar

Extract the time and event variable from a Cox model

Description

Extract the time and event variable from a Cox model

Usage

SurvResponseVar(formula)

Arguments

formula a formula

Author(s)

Brice Ozenne broz@sund.ku.dk

Examples

## Not run:
SurvResponseVar(Surv(time,event)~X1+X2)
SurvResponseVar(Hist(time,event==0)~X1+X2)
SurvResponseVar(Surv(start,time, status,type="counting") ~ X3+X5)
SurvResponseVar(Surv(start,event=status, time2=time,type="counting") ~ X3+X5)
SurvResponseVar(survival::Surv(start,event=status, time2=time,type="counting") ~ X3+X5)
SurvResponseVar(I(status == 1) ~ X3+X5)
SurvResponseVar(list(Hist(time, event) ~ X1+X6,Hist(time, event) ~ X6))

## End(Not run)

terms.phreg

Extract terms for phreg objects

Description

Extract terms for phreg objects

Usage

## S3 method for class 'phreg'
terms(x, ...)

## Not run:
SurvResponseVar(Surv(time,event)~X1+X2)
SurvResponseVar(Hist(time,event==0)~X1+X2)
SurvResponseVar(Surv(start,time, status,type="counting") ~ X3+X5)
SurvResponseVar(Surv(start,event=status, time2=time,type="counting") ~ X3+X5)
SurvResponseVar(survival::Surv(start,event=status, time2=time,type="counting") ~ X3+X5)
SurvResponseVar(I(status == 1) ~ X3+X5)
SurvResponseVar(list(Hist(time, event) ~ X1+X6,Hist(time, event) ~ X6))

## End(Not run)
transformCIBP

Arguments

- **x**: a phreg object.
- **...**: not used.

**transformCI**  
*Compute Confidence Intervals using a transformation*

**Description**

Compute confidence intervals using a transformation. The resulting confidence interval is returned on the original case (i.e. back-transformed).

**Usage**

```r
transformCI(estimate, se, quantile, type, min.value, max.value)
```

**Arguments**

- **estimate**: [numeric matrix] the estimate value before transformation.
- **se**: [numeric matrix] the standard error after transformation.
- **quantile**: [numeric vector] quantile that will be multiplied to each column of se.
- **type**: [character] the transformation. Can be "log", "loglog", "cloglog", or "atanh" (Fisher transform).
- **min.value**: [numeric] if not NULL and the lower bound of the confidence interval is below min, it will be set at min.
- **max.value**: [numeric] if not NULL and the lower bound of the confidence interval is below max, it will be set at max.

**Details**

se and estimate must have same dimensions.

**transformCIBP**  
*Compute Confidence Intervals/Bands and P-values After a Transformation*

**Description**

Compute confidence intervals/bands and p-values after a transformation
transformCIBP

Usage

transformCIBP(
    estimate,
    se,
    iid,
    null,
    conf.level,
    nsim.band,
    seed,
    type,
    min.value,
    max.value,
    ci,
    band,
    p.value
)

Arguments

estimate        [numeric matrix] the estimate value before transformation.
se              [numeric matrix] the standard error before transformation.
IID             [numeric array] the iid decomposition before transformation.
null            [numeric] the value of the estimate (before transformation) under the null hypothesis.
conf.level      [numeric, 0-1] Level of confidence.
nsim.band       [integer, >0] the number of simulations used to compute the quantiles for the confidence bands.
seed            [integer, >0] seed number set before performing simulations for the confidence bands.
type            [character] the transformation. Can be "log", "loglog", "cloglog", or "atanh" (Fisher transform).
min.value       [numeric] if not NULL and the lower bound of the confidence interval is below min, it will be set at min.
max.value       [numeric] if not NULL and the lower bound of the confidence interval is below max, it will be set at max.
ci              [logical] should confidence intervals be computed.
band            [logical] should confidence bands be computed.
p.value         [logical] should p-values be computed.

The IID decomposition must have dimensions [n.prediction,time,n.obs] while estimate and se must have dimensions [n.prediction,time].
**transformIID**

*Compute Influence Functions after Transformation*

**Description**

Compute influence functions after transformation based on the influence function before transformation.

**Usage**

transformIID(estimate, iid, type)

**Arguments**

- **estimate** [numeric matrix] the estimate value before transformation.
- **iid** [numeric array] the standard error before transformation.
- **type** [character] the transformation. Can be "log", "loglog", "cloglog", or "atanh" (Fisher transform).

**Details**

Use a delta method to find the standard error after transformation.

The iid decomposition must contain have dimension [n.prediction,time,n.obs] and estimate [n.prediction,time].

**transformP**

*Compute P-values After a Transformation*

**Description**

Compute the p-values after a transformation.

**Usage**

transformP(estimate, se, null, type)

**Arguments**

- **estimate** [numeric matrix] the estimate value before transformation.
- **se** [numeric matrix] the standard error after transformation.
- **null** [numeric] the value of the estimate (before transformation) under the null hypothesis.
- **type** [character] the transformation. Can be "log", "loglog", "cloglog", or "atanh" (Fisher transform).
transformSE

Details

se and estimate must have same dimensions.

---

transformSE

Compute Standard Errors after Transformation

Description

Compute standard errors after transformation based on the standard error before transformation.

Usage

transformSE(estimate, se, type)

Arguments

- **estimate** [numeric matrix] the estimate value before transformation.
- **se** [numeric matrix] the standard error before transformation.
- **type** [character] the transformation. Can be "log", "loglog", "cloglog", or "atanh" (Fisher transform).

Details

Use a delta method to find the standard error after transformation.

se and estimate must have same dimensions.
Index

*Topic **datasets**
   Melanoma, 64
   Paquid, 66
*Topic **survival**
   Cforest, 25
   CSC, 46
   FGR, 50
   ipcw, 61
   plot.riskRegression, 68
   plotEffects, 75
   plotPredictRisk, 77
   predict.riskRegression, 87
   predictRisk, 93
   riskRegression, 107
   selectCox, 125
   subjectWeights, 130
   ARR (**riskRegression**), 107
   as.data.table.ate, 4
   as.data.table.influenceTest, 5
   as.data.table.predictCox, 5
   as.data.table.predictCSC, 6
   ate, 6, 13, 99, 133
   autoplot.ate, 8, 12
   autoplot.predictCox, 13, 90
   autoplot.predictCSC, 15, 86
   autoplot.Score, 16
   axis, 78
   boot2pvalue, 18
   boxplot.Score, 19
   calcSeCox, 21
   calcSeCSC, 23
   Cforest, 25
   coef.CauseSpecificCox, 26
   coef.riskRegression, 26
   colCenter_cpp, 27
   colCumProd, 27
   colCumSum, 28
   colMultiply_cpp, 29
   colScale_cpp, 29
   colSumsCrossprod, 30
   confBandCox, 31
   confint.ate, 8, 31, 99, 133
   confint.influenceTest, 33, 101
   confint.predictCox, 34, 90, 102
   confint.predictCSC, 36, 86, 103
   coxBaseEstimator, 38
   coxCenter, 38
   coxFormula, 39
   coxLP, 40
   coxModelFrame, 40
   coxN, 41
   coxph, 47
   coxSpecial, 42
   coxStrata, 43
   coxStrataLevel, 44
   coxVarCov, 44
   coxVariableName, 45
   CSC, 24, 46
   Ctree, 49
   discreteRoot, 50
   fastbw, 125
   FGR, 50
   getSplitMethod, 52
   Hist, 50
   iidCox, 53
   influenceTest, 56, 101
   IPA, 59
   ipcw, 61
   legend, 78
   lines, 74
   LRR (**riskRegression**), 107

142