Package 'pec'

October 11, 2021

Title Prediction Error Curves for Risk Prediction Models in Survival Analysis

Version 2021.10.11

Author Thomas A. Gerds

Description Validation of risk predictions obtained from survival models and competing risk models based on censored data using inverse weighting and cross-validation. Most of the 'pec' functionality has been moved to 'riskRegression'.

Depends R (>= 2.9.0), prodlim (>= 1.4.9)

Imports foreach (>= 1.4.2), rms (>= 4.2-0), survival (>= 2.37-7), riskRegression (>= 2020.02.05), lava (>= 1.4.1), timereg (>= 1.8.9),

Suggests randomForestSRC, party, cmprsk (>= 2.2-7), rpart, crrstep, Hmisc (>= 3.14-4)

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

License GPL (>= 2)

RoxygenNote 7.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-10-11 14:30:02 UTC

R topics documented:

`calPlot` *Calibration plots for right censored data*

Description

Calibration plots for risk prediction models in right censored survival and competing risks data

Usage

```
calPlot(
  object,
  time,
  formula,
  data,
  splitMethod = "none",
  B = 1,
  M,
  pseudo,
  type,
  showPseudo,
  pseudo.col = NULL,
  pseudo.pch = NULL,
  method = "nne",
  round = TRUE,
  bandwidth = NULL,
  q = 10,
```

```
  bars = FALSE,
  hanging = FALSE,
  names = "quantiles",
  showFrequencies = FALSE,
  jack.density = 55,
  plot = TRUE,
  add = FALSE,
  diag = !add,
  legend = !add,
  axes = !add,
  xlim = c(0, 1),
  ylim = c(0, 1),
  xlab,
  ylab,
  col,
  lwd,
  lty,
  pch,
  cause = 1,
  percent = TRUE,
  giveToModel = NULL,
  na.action = na.fail,
  cores = 1,
  verbose = FALSE,
  cex = 1,
  ...
)
```

Arguments

`object`	A named list of prediction models, where allowed entries are (1) R-objects for which a predictSurvProb method exists (see details), (2) a `call` that evaluates to such an R-object (see examples), (3) a matrix with predicted probabilities having as many rows as `data` and as many columns as `times`. For cross-validation all objects in this list must include their `call`.
`time`	The evaluation time point at predicted event probabilities are plotted against pseudo-observed event status.
`formula`	A survival or event history formula. The left hand side is used to compute the expected event status. If `formula` is `missing`, try to extract a formula from the first element in object.
`data`	A data frame in which to validate the prediction models and to fit the censoring model. If `data` is missing, try to extract a data set from the first element in object.
`splitMethod`	Defines the internal validation design: `none/noPlan`: Assess the models in the give `data`, usually either in the same data where they are fitted, or in independent test data. `BootCv`: Bootstrap cross validation. The prediction models are trained on `B` bootstrap samples, that are either drawn with replacement of the same size as

	the original data or without replacement from `data` of the size `M`. The models are assessed in the observations that are NOT in the bootstrap sample.
`B`	The number of cross-validation steps.
`M`	The size of the subsamples for cross-validation.
`pseudo`	Logical. Determines the method for estimating expected event status: `TRUE`: Use average pseudo-values. `FALSE`: Use the product-limit estimate, i.e., apply the Kaplan-Meier method for right censored survival and the Aalen-Johansen method for right censored competing risks data.
`type`	Either "risk" or "survival".
`showPseudo`	If `TRUE` the pseudo-values are shown as dots on the plot (only when `pseudo=TRUE`).
`pseudo.col`	Colour for pseudo-values.
`pseudo.pch`	Dot type (see par) for pseudo-values.
`method`	The method for estimating the calibration curve(s): `"nne"`: The expected event status is obtained in the nearest neighborhood around the predicted event probabilities. `"quantile"`: The expected event status is obtained in groups defined by quantiles of the predicted event probabilities.
`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`.
`showFrequencies`	Barplots only. If `TRUE`, show frequencies above the bars.
`jack.density`	Gray scale for pseudo-observations.
`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`	If `FALSE` no legend is drawn.
`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 points.
cause	For competing risks models, the cause of failure or event of interest
percent	If TRUE axes labels are multiplied by 100 and thus interpretable on a percent scale.
giveToModel	List of with exactly one entry for each entry in object. Each entry names parts of the value of the fitted models that should be extracted and added to the value.
na.action	Passed to model.frame
cores	Number of cores for parallel computing. Passed as value of argument mc.cores to mclapply.
verbose	if TRUE report details of the progress, e.g. count the steps in cross-validation.
cex	Default cex used for legend and labels.
...	Used to control the subroutines: plot, axis, lines, barplot, legend. See SmartControl.

Details

For method "nne" the optimal bandwidth with respect to is obtained with the function dpik from the package KernSmooth for a box kernel function.

Value

list with elements: time, pseudoFrame and bandwidth (NULL for method quantile).

Author(s)

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

Examples

```
library(prodlim)
library(lava)
library(riskRegression)
library(survival)
# survival
dlearn <- SimSurv(40)
dval <- SimSurv(100)
f <- coxph(Surv(time,status)~X1+X2,data=dlearn,x=TRUE,y=TRUE)
cf=calPlot(f,time=3,data=dval)
print(cf)
plot(cf)

g <- coxph(Surv(time,status)~X2,data=dlearn,x=TRUE,y=TRUE)
cf2=calPlot(list("Cox regression X1+X2"=f,"Cox regression X2"=g),
    time=3,
    type="risk",
    data=dval)
print(cf2)
```

```
plot(cf2)
calPlot(f,time=3,data=dval,type="survival")
calPlot(f,time=3,data=dval,bars=TRUE,pseudo=FALSE)
calPlot(f,time=3,data=dval,bars=TRUE,type="risk",pseudo=FALSE)

## show a red line which follows the hanging bars
calPlot(f,time=3,data=dval,bars=TRUE,hanging=TRUE)
a <- calPlot(f,time=3,data=dval,bars=TRUE,hanging=TRUE,abline.col=NULL)
lines(c(0,1,ceiling(a$xcoord)),
      c(a$offset[1],a$offset,a$offset[length(a$offset)]),
      col=2,lwd=5,type="s")

calPlot(f,time=3,data=dval,bars=TRUE,type="risk",hanging=TRUE)

set.seed(13)
m <- crModel()
regression(m, from = "X1", to = "eventtime1") <- 1
regression(m, from = "X2", to = "eventtime1") <- 1
m <- addvar(m,c("X3","X4","X5"))
distribution(m, "X1") <- binomial.lvm()
distribution(m, "X4") <- binomial.lvm()
d1 <- sim(m,100)
d2 <- sim(m,100)
csc <- CSC(Hist(time,event)~X1+X2+X3+X4+X5,data=d1)
fgr <- FGR(Hist(time,event)~X1+X2+X3+X4+X5,data=d1,cause=1)
if ((requireNamespace("cmprsk",quietly=TRUE))){
predict.crr <- cmprsk:::predict.crr
cf3=calPlot(list("Cause-specific Cox"=csc,"Fine-Gray"=fgr),
        time=5,
        legend.x=-0.3,
        legend.y=1.35,
        ylab="Observed event status",
        legend.legend=c("Cause-specific Cox regression","Fine-Gray regression"),
        legend.xpd=NA)
print(cf3)
plot(cf3)

b1 <- calPlot(list("Fine-Gray"=fgr),time=5,bars=TRUE,hanging=FALSE)
print(b1)
plot(b1)

calPlot(fgr,time=5,bars=TRUE,hanging=TRUE)
}
```

---

cindex *Concordance index for right censored survival time data*

Description

In survival analysis, a pair of patients is called concordant if the risk of the event predicted by a model is lower for the patient who experiences the event at a later timepoint. The concordance probability (C-index) is the frequency of concordant pairs among all pairs of subjects. It can be used to measure and compare the discriminative power of a risk prediction models. The function provides an inverse of the probability of censoring weigthed estimate of the concordance probability to adjust for right censoring. Cross-validation based on bootstrap resampling or bootstrap subsampling can be applied to assess and compare the discriminative power of various regression modelling strategies on the same set of data.

Usage

```
cindex(
  object,
  formula,
  data,
  eval.times,
  pred.times,
  cause,
  lyl = FALSE,
  cens.model = "marginal",
  ipcw.refit = FALSE,
  ipcw.args = NULL,
  ipcw.limit,
  tiedPredictionsIn = TRUE,
  tiedOutcomeIn = TRUE,
  tiedMatchIn = TRUE,
  splitMethod = "noPlan",
  B,
  M,
  model.args = NULL,
  model.parms = NULL,
  keep.models = FALSE,
  keep.residuals = FALSE,
  keep.pvalues = FALSE,
  keep.weights = FALSE,
  keep.index = FALSE,
  keep.matrix = FALSE,
  multiSplitTest = FALSE,
  testTimes,
  confInt = FALSE,
  confLevel = 0.95,
  verbose = TRUE,
  savePath = NULL,
  slaveseed = NULL,
  na.action = na.fail,
  ...
)
```

Arguments

object	A named list of prediction models, where allowed entries are (1) R-objects for which a predictSurvProb method exists (see details), (2) a call that evaluates to such an R-object (see examples), (3) a matrix with predicted probabilities having as many rows as data and as many columns as times. For cross-validation all objects in this list must include their call.
formula	A survival formula. The left hand side is used to finde the status response variable in data. For right censored data, the right hand side of the formula is used to specify conditional censoring models. For example, set Surv(time,status)~x1+x2 and cens.model="cox". Then the weights are based on a Cox regression model for the censoring times with predictors x1 and x2. Note that the usual coding is assumed: status=0 for censored times and that each variable name that appears in formula must be the column name in data. If there are no covariates, i.e. formula=Surv(time,status)~1 the cens.model is coerced to "marginal" and the Kaplan-Meier estimator for the censoring times is used to calculate the weights. If formula is missing, try to extract a formula from the first element in object.
data	A data frame in which to validate the prediction models and to fit the censoring model. If data is missing, try to extract a data set from the first element in object.
eval.times	A vector of timepoints for evaluating the discriminative ability. At each timepoint the c-index is computed using only those pairs where one of the event times is known to be earlier than this timepoint. If eval.times is missing then the largest uncensored event time is used.
pred.times	A vector of timepoints for evaluating the prediction models. This should either be exactly one timepoint used for all eval.times, or be as long as eval.times, in which case the predicted order of risk for the jth entry of eval.times is based on the jth entry of pred.times corresponding
cause	For competing risks, the event of interest. Defaults to the first state of the response, which is obtained by evaluating the left hand side of formula in data.
lyl	If TRUE rank subjects accoring to predicted life-years-lost (See Andersen due to this cause instead of predicted risk.
cens.model	Method for estimating inverse probability of censoring weigths: cox: A semi-parametric Cox proportional hazard model is fitted to the censoring times marginal: The Kaplan-Meier estimator for the censoring times nonpar: Nonparametric extension of the Kaplan-Meier for the censoring times using symmetric nearest neighborhoods – available for arbitrary many strata variables on the right hand side of argument formula but at most one continuous variable. See the documentation of the functions prodlim and neighborhood from the prodlim package. aalen: The nonparametric Aalen additive model fitted to the censoring times. Requires the timereg package maintained by Thomas Scheike.
ipcw.refit	If TRUE the inverse probability of censoring weigths are estimated separately in each training set during cross-validation.

ipcw.args — List of arguments passed to function specified by argument cens.model.

ipcw.limit — Value between 0 and 1 (but no equal to 0!) used to cut for small weights in order to stabilize the estimate at late times were few individuals are observed.

tiedPredictionsIn — If FALSE pairs with identical predictions are excluded, unless also the event times are identical and uncensored and tiedMatchIn is set to TRUE.

tiedOutcomeIn — If TRUE pairs with identical and uncensored event times are excluded, unless also the predictions are identical and tiedMatchIn is set to TRUE.

tiedMatchIn — If TRUE then pairs with identical predictions and identical and uncensored event times are counted as concordant pairs.

splitMethod — Defines the internal validation design:

none/noPlan: Assess the models in the give data, usually either in the same data where they are fitted, or in independent test data.

BootCv: Bootstrap cross validation. The prediction models are trained on B bootstrap samples, that are either drawn with replacement of the same size as the original data or without replacement from data of the size M. The models are assessed in the observations that are NOT in the bootstrap sample.

Boot632: Linear combination of AppCindex and OutOfBagCindex using the constant weight .632.

B — Number of bootstrap samples. The default depends on argument splitMethod. When splitMethod in c("BootCv","Boot632") the default is 100. For splitMethod="none" B is the number of bootstrap simulations e.g. to obtain bootstrap confidence limits – default is 0.

M — The size of the bootstrap samples for resampling without replacement. Ignored for resampling with replacement.

model.args — List of extra arguments that can be passed to the predictSurvProb methods. The list must have an entry for each entry in object.

model.parms — Experimental. List of with exactly one entry for each entry in object. Each entry names parts of the value of the fitted models that should be extracted and added to the value.

keep.models — Logical. If TRUE keep the models in object. Since fitted models can be large objects the default is FALSE.

keep.residuals — Experimental.

keep.pvalues — Experimental.

keep.weights — Experimental.

keep.index — Logical. If FALSE remove the bootstrap or cross-validation index from the output list which otherwise is included in the method part of the output list.

keep.matrix — Logical. If TRUE add all B prediction error curves from bootstrapping or cross-validation to the output.

multiSplitTest — Experimental.

testTimes — A vector of time points for testing differences between models in the time-point specific Brier scores.

confInt — Experimental.

confLevel	Experimental.
verbose	if TRUE report details of the progress, e.g. count the steps in cross-validation.
savePath	Place in your filesystem (directory) where training models fitted during cross-validation are saved. If missing training models are not saved.
slaveseed	Vector of seeds, as long as B, to be given to the slaves in parallel computing.
na.action	Passed immediately to model.frame. Defaults to na.fail. If set otherwise most prediction models will not work.
...	Not used.

Details

Pairs with identical observed times, where one is uncensored and one is censored, are always considered usuable (independent of the value of tiedOutcomeIn), as it can be assumed that the event occurs at a later timepoint for the censored observation.

For uncensored response the result equals the one obtained with the functions rcorr.cens and rcorrcens from the Hmisc package (see examples).

Value

Estimates of the C-index.

Author(s)

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

References

TA Gerds, MW Kattan, M Schumacher, and C Yu. Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring. Statistics in Medicine, Ahead of print:to appear, 2013. DOI = 10.1002/sim.5681

Wolbers, M and Koller, MT and Witteman, JCM and Gerds, TA (2013) Concordance for prognostic models with competing risks Research report 13/3. Department of Biostatistics, University of Copenhagen

Andersen, PK (2012) A note on the decomposition of number of life years lost according to causes of death Research report 12/2. Department of Biostatistics, University of Copenhagen

Paul Blanche, Michael W Kattan, and Thomas A Gerds. The c-index is not proper for the evaluation of-year predicted risks. Biostatistics, 20(2): 347–357, 2018.

Examples

```
 # simulate data based on Weibull regression
library(prodlim)
 set.seed(13)
 dat <- SimSurv(100)
 # fit three different Cox models and a random survival forest
 # note: low number of trees for the purpose of illustration
```

```
 library(survival)
 library(randomForestSRC)
 cox12 <- coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE)
 cox1 <- coxph(Surv(time,status)~X1,data=dat,x=TRUE,y=TRUE)
 cox2 <- coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE)
 rsf1 <- rfsrc(Surv(time,status)~X1+X2,data=dat,ntree=15,forest=TRUE)
 #
 # compute the apparent estimate of the C-index at different time points
 #
A1  <- pec::cindex(list("Cox X1"=cox1,
                     "RSF"=rsf1),
  formula=Surv(time,status)~X1+X2,
  data=dat,
  eval.times=10)
ApparrentCindex  <- pec::cindex(list("Cox X1"=cox1,
       "Cox X2"=cox2,
       "Cox X1+X2"=cox12,
                     "RSF"=rsf1),
  formula=Surv(time,status)~X1+X2,
  data=dat,
  eval.times=seq(1,15,1))
  print(ApparrentCindex)
  plot(ApparrentCindex)
 #
 # compute the bootstrap-crossvalidation estimate of
 # the C-index at different time points
 #
set.seed(142)
bcvCindex  <- pec::cindex(list("Cox X1"=cox1,
       "Cox X2"=cox2,
       "Cox X1+X2"=cox12,
                     "RSF"=rsf1),
  formula=Surv(time,status)~X1+X2,
  data=dat,
                 splitMethod="bootcv",
                 B=5,
    eval.times=seq(1,15,1))
  print(bcvCindex)
  plot(bcvCindex)
 # for uncensored data the results are the same
 # as those obtained with the function rcorr.cens from Hmisc

set.seed(16)
dat <- SimSurv(30)
dat$staus=1
fit12 <- coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE)
fit1 <- coxph(Surv(time,status)~X1,data=dat,x=TRUE,y=TRUE)
fit2 <- coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE)
Cpec <- pec::cindex(list("Cox X1+X2"=fit12,"Cox X1"=fit1,"Cox X2"=fit2),
       formula=Surv(time,status)~1,
       data=dat)
p1 <- predictSurvProb(fit1,newdata=dat,times=10)
p2 <- predictSurvProb(fit2,newdata=dat,times=10)
```

```
p12 <- predictSurvProb(fit12,newdata=dat,times=10)
if (requireNamespace("Hmisc",quietly=TRUE)){
library(Hmisc)
harrelC1 <- rcorr.cens(p1,with(dat,Surv(time,status)))
harrelC2 <- rcorr.cens(p2,with(dat,Surv(time,status)))
harrelC12 <- rcorr.cens(p12,with(dat,Surv(time,status)))
harrelC1[["C Index"]]==Cpec$AppCindex[["Cox.X1"]]
harrelC2[["C Index"]]==Cpec$AppCindex[["Cox.X2"]]
harrelC12[["C Index"]]==Cpec$AppCindex[["Cox.X1.X2"]]
}
 #
 # competing risks
 #
library(riskRegression)
library(prodlim)
set.seed(30)
dcr.learn <- SimCompRisk(30)
dcr.val <- SimCompRisk(30)
pec::cindex(CSC(Hist(time,event)~X1+X2,data=dcr.learn),data=dcr.val)
fit <- CSC(Hist(time,event)~X1+X2,data=dcr.learn)
cif <- predictRisk(fit,newdata=dcr.val,times=3,cause=1)
pec::cindex(list(fit),data=dcr.val,times=3)
```

`cost` *Copenhagen Stroke Study*

Description

This data set contains a subset of the data from the Copenhagen stroke study.

Format

This data frame contains the observations of 518 stroke patients :

age Age of the patients in years.

sex A factor with two levels `female` and `male`.

hypTen Hypertension, a factor with two levels `no` and `yes`.

ihd History of ischemic heart disease at admission, a factor with two levels `no` and `yes`.

prevStroke History of previous strokes before admission, a factor with two levels `no` and `yes`.

othDisease History of other disabling diseases (e.g. severe dementia), a factor with two levels `no` and `yes`.

alcohol Daily alcohol consumption, a factor with two levels `no` and `yes`.

diabetes Diabetes mellitus status indicating if the glucose level was higher than 11 mmol/L, a factor with two levels `no` and `yes`.

smoke Daily smoking status, a factor with two levels `no` and `yes`.

atrialFib Atrial fibrillation, a factor with two levels `no` and `yes`.

hemor Hemorrhage (stroke subtype), a factor with two levels no (infarction) and yes (hemorrhage).

strokeScore Scandinavian stroke score at admission to the hospital. Ranges from 0 (worst) to 58 (best).

cholest Cholesterol level

time Survival time (in days).

status Status (0: censored, 1: event).

References

Joergensen HS, Nakayama H, Reith J, Raaschou HO, and Olsen TS. Acute stroke with atrial fibrillation. The Copenhagen Stroke Study. Stroke, 27(10):1765-9, 1996.

Mogensen UB, Ishwaran H, and Gerds TA. Evaluating random forests for survival analysis using prediction error curves. Technical Report 8, University of Copenhagen, Department of Biostatistics, 2010.

---

coxboost *Formula interface for function* CoxBoost *of package* CoxBoost.

Description

Formula interface for function CoxBoost of package CoxBoost.

Usage

```
coxboost(formula, data, cv = TRUE, cause = 1, penalty, ...)
```

Arguments

formula	An event-history formula for competing risks of the form Hist(time,status)~sex+age where status defines competing events and right censored data. The code for right censored can be controlled with argument cens.code, see man page the function Hist.
data	A data.frame in which the variables of formula are defined.
cv	If TRUE perform cross-validation to optimize the parameter stepno. This calls the function cv.CoxBoost whose arguments are prefix controlled, that is cv.K=7 sets the argument K of cv.CoxBoost to 7. If FALSE use stepno.
cause	The cause of interest in competing risk models.
penalty	See CoxBoost.
...	Arguments passed to either CoxBoost via CoxBoost.arg or to cv.CoxBoost via cv.CoxBoost.arg.

Details

See CoxBoost.

Value

See CoxBoost.

Author(s)

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

References

See CoxBoost.

See Also

See CoxBoost.

---

crps *Summarizing prediction error curves*

Description

Computes the cumulative prediction error curves, aka integrated Brier scores, in ranges of time.

Usage

```
crps(object, models, what, times, start)
```

Arguments

object	An object with estimated prediction error curves obtained with the function pec
models	Which models in object$models should be considered.
what	The name of the entry in x to be cumulated. Defauls to PredErr Other choices are AppErr, BootCvErr, Boot632, Boot632plus.
times	Time points at which the integration of the prediction error curve stops.
start	The time point at which the integration of the prediction error curve is started.

Details

The cumulative prediction error (continuous ranked probability score) is defined as the area under the prediction error curve, hence the alias name, ibs, which is short for integrated Brier score.

Value

A matrix with a column for the crps (ibs) at every requested time point and a row for each model

Author(s)

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

References

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

Gerds TA, Cai T & Schumacher M (2008) The performance of risk prediction models Biometrical Journal, 50(4), 457–479

See Also

pec

Examples

```
set.seed(18713)
library(prodlim)
library(survival)
dat=SimSurv(100)
pmodel=coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE)
perror=pec(list(Cox=pmodel),Hist(time,status)~1,data=dat)

## cumulative prediction error
crps(perror,times=1) # between min time and 1
## same thing:
ibs(perror,times=1) # between min time and 1
crps(perror,times=1,start=0) # between 0 and 1
crps(perror,times=seq(0,1,.2),start=0) # between 0 and seq(0,1,.2)
```

GBSG2 *German Breast Cancer Study Group 2*

Description

A data frame containing the observations from the GBSG2 study.

Format

This data frame contains the observations of 686 women:

horTh hormonal therapy, a factor at two levels `no` and `yes`.

age of the patients in years.

menostat menopausal status, a factor at two levels `pre` (premenopausal) and `post` (postmenopausal).

tsize tumor size (in mm).

tgrade tumor grade, a ordered factor at levels `I` < `II` < `III`.

pnodes number of positive nodes.

progrec progesterone receptor (in fmol).

estrec estrogen receptor (in fmol).

time recurrence free survival time (in days).

cens censoring indicator (0- censored, 1- event).

References

M. Schumacher, G. Basert, H. Bojar, K. Huebner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R.L.A. Neumann and H.F. Rauschecker for the German Breast Cancer Study Group (1994), Randomized 2×2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*, **12**, 2086–2093.

`ipcw` *Estimation of censoring probabilities*

Description

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

Usage

```
ipcw(
  formula,
  data,
  method,
  args,
  times,
  subjectTimes,
  subjectTimesLag = 1,
  what
)
```

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

subjectTimesLag
: If equal to 1 then obtain G(T_i-|X_i), if equal to 0 estimate the conditional censoring distribution at the subjectTimes, 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.subjectTimes" then weights are estimated at individual subjectTimes. If missing then produce both.

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.subjectTimes in the right order for some choices of method.

Value

times
: The times at which weights are estimated

IPCW.times
: Estimated weights at times

IPCW.subjectTimes
: Estimated weights at individual time values subjectTimes

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>

See Also

pec

Examples

```
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)),
```

```
  subjectTimes=dat$time)
plot(WKM$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$time)),
  subjectTimes=dat$time)
WCox$fit

plot(WKM$fit)
lines(sort(unique(dat$time)),
      1-WCox$IPCW.times[1,],
      type="l",
      col=2,
      lty=3,
      lwd=3)
lines(sort(unique(dat$time)),
      1-WCox$IPCW.times[5,],
      type="l",
      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$time)),
  subjectTimes=dat$time)
plot(WKM2$fit,add=FALSE)
```

Pbc3 *Pbc3 data*

Description

PBC3 was a multi-centre randomized clinical trial conducted in six European hospitals. Between 1 Jan. 1983 and 1 Jan. 1987, 349 patients with the liver disease primary biliary cirrhosis (PBC) were randomized to either treatment with Cyclosporin A (CyA, 176 patients) or placebo (173 patients). The purpose of the trial was to study the effect of treatment on the survival time. However, during the course of the trial an increased use of liver transplantation for patients with this disease made the investigators redefine the main response variable to be time to "failure of medical treatment"

defined as either death or liver transplantation. Patients were then followed from randomization until treatment failure, drop-out or 1 Jan, 1989; 61 patients died (CyA: 30, placebo: 31), another 29 were transplanted (CyA: 14, placebo: 15) and 4 patients were lost to follow-up before 1 Jan. 1989. At entry a number of clinical, biochemical and histological variables, including serum bilirubin, serum albumin, sex, age were recorded.

Format

A data frame with 349 observations on the following 15 variables.

ptno patient identification

unit hospital (1: Hvidovre, 2: London, 3: Copenhagen, 4: Barcelona, 5: Munich, 6: Lyon)

tment treatment (0: placebo, 1: CyA)

sex (1: males, 0: females)

age age in years

stage histological stage (1, 2, 3, 4)

gibleed previous gastrointestinal bleeding (1: yes, 0: no)

crea creatinine (micromoles/L)

alb albumin (g/L)

bili bilirubin (micromoles/L)

alkph alkaline phosphatase (IU/L)

asptr aspartate transaminase (IU/L)

weight body weight (kg)

days observation time (days)

status status at observation time (0: censored, 1: liver transplantation, 2 : dead)

Source

Andersen and Skovgaard. Regression with linear predictors.

References

Andersen and Skovgaard. Regression with linear predictors. Springer, 2010.

Examples

```
data(Pbc3)
```

pec *Prediction error curves*

Description

Evaluating the performance of risk prediction models in survival analysis. The Brier score is a weighted average of the squared distances between the observed survival status and the predicted survival probability of a model. Roughly the weights correspond to the probabilities of not being censored. The weights can be estimated depend on covariates. Prediction error curves are obtained when the Brier score is followed over time. Cross-validation based on bootstrap resampling or bootstrap subsampling can be applied to assess and compare the predictive power of various regression modelling strategies on the same set of data.

Usage

```
pec(
  object,
  formula,
  data,
  traindata,
  times,
  cause,
  start,
  maxtime,
  exact = TRUE,
  exactness = 100,
  fillChar = NA,
  cens.model = "cox",
  ipcw.refit = FALSE,
  ipcw.args = NULL,
  splitMethod = "none",
  B,
  M,
  reference = TRUE,
  model.args = NULL,
  model.parms = NULL,
  keep.index = FALSE,
  keep.matrix = FALSE,
  keep.models = FALSE,
  keep.residuals = FALSE,
  keep.pvalues = FALSE,
  noinf.permute = FALSE,
  multiSplitTest = FALSE,
  testIBS,
  testTimes,
  confInt = FALSE,
  confLevel = 0.95,
```

```
  verbose = TRUE,
  savePath = NULL,
  slaveseed = NULL,
  na.action = na.fail,
  ...
)
```

Arguments

object	A named list of prediction models, where allowed entries are (1) R-objects for which a predictSurvProb method exists (see details), (2) a call that evaluates to such an R-object (see examples), (3) a matrix with predicted probabilities having as many rows as data and as many columns as times. For cross-validation all objects in this list must include their call.
formula	A survival formula as obtained either with prodlim::Hist or survival::Surv. The left hand side is used to find the status response variable in data. For right censored data, the right hand side of the formula is used to specify conditional censoring models. For example, set Surv(time,status)~x1+x2 and cens.model="cox". Then the weights are based on a Cox regression model for the censoring times with predictors x1 and x2. Note that the usual coding is assumed: status=0 for censored times and that each variable name that appears in formula must be the column name in data. If there are no covariates, i.e. formula=Surv(time,status)~1 the cens.model is coerced to "marginal" and the Kaplan-Meier estimator for the censoring times is used to calculate the weights. If formula is missing, try to extract a formula from the first element in object.
data	A data frame in which to validate the prediction models and to fit the censoring model. If data is missing, try to extract a data set from the first element in object.
traindata	A data frame in which the models are trained. This argument is used only in the absence of crossvalidation, in which case it is passed to the predictHandler function predictSurvProb
times	A vector of time points. At each time point the prediction error curves are estimated. If exact==TRUE the times are merged with all the unique values of the response variable. If times is missing and exact==TRUE all the unique values of the response variable are used. If missing and exact==FALSE use a equidistant grid of values between start and maxtime. The distance is determined by exactness.
cause	For competing risks, the event of interest. Defaults to the first state of the response, which is obtained by evaluating the left hand side of formula in data.
start	Minimal time for estimating the prediction error curves. If missing and formula defines a Surv or Hist object then start defaults to 0, otherwise to the smallest observed value of the response variable. start is ignored if times are given.
maxtime	Maximal time for estimating the prediction error curves. If missing the largest value of the response variable is used.

exact — Logical. If TRUE estimate the prediction error curves at all the unique values of the response variable. If times are given and exact=TRUE then the times are merged with the unique values of the response variable.

exactness — An integer that determines how many equidistant gridpoints are used between start and maxtime. The default is 100.

fillChar — Symbol used to fill-in places where the values of the prediction error curves are not available. The default is NA.

cens.model — Method for estimating inverse probability of censoring weigths:

cox: A semi-parametric Cox proportional hazard model is fitted to the censoring times

marginal: The Kaplan-Meier estimator for the censoring times

nonpar: Nonparametric extension of the Kaplan-Meier for the censoring times using symmetric nearest neighborhoods – available for arbitrary many strata variables on the right hand side of argument formula but at most one continuous variable. See the documentation of the functions prodlim and neighborhood from the prodlim package.

aalen: The nonparametric Aalen additive model fitted to the censoring times. Requires the timereg package.

ipcw.refit — If TRUE the inverse probability of censoring weigths are estimated separately in each training set during cross-validation.

ipcw.args — List of arguments passed to function specified by argument cens.model.

splitMethod — SplitMethod for estimating the prediction error curves.

none/noPlan: Assess the models in the same data where they are fitted. boot: DEPRECIATED.

cvK: K-fold cross-validation, i.e. cv10 for 10-fold cross-validation. After splitting the data in K subsets, the prediction models (ie those specified in object) are evaluated on the data omitting the Kth subset (training step). The prediction error is estimated with the Kth subset (validation step).

The random splitting is repeated B times and the estimated prediction error curves are obtained by averaging.

BootCv: Bootstrap cross validation. The prediction models are trained on B bootstrap samples, that are either drawn with replacement of the same size as the original data or without replacement from data of the size M. The models are assessed in the observations that are NOT in the bootstrap sample.

Boot632: Linear combination of AppErr and BootCvErr using the constant weight .632.

Boot632plus: Linear combination of AppErr and BootCv using weights dependent on how the models perform in permuted data.

loocv: Leave one out cross-validation.

NoInf: Assess the models in permuted data.

B — Number of bootstrap samples. The default depends on argument splitMethod. When splitMethod in c("BootCv","Boot632","Boot632plus") the default is 100. For splitMethod="cvK" B is the number of cross-validation cycles, and – default is 1. For splitMethod="none" B is the number of bootstrap simulations e.g. to obtain bootstrap confidence limits – default is 0.

M	The size of the bootstrap samples for resampling without replacement. Ignored for resampling with replacement.
reference	Logical. If TRUE add the marginal Kaplan-Meier prediction model as a reference to the list of models.
model.args	List of extra arguments that can be passed to the predictSurvProb methods. The list must have an entry for each entry in object.
model.parms	Experimental. List of with exactly one entry for each entry in object. Each entry names parts of the value of the fitted models that should be extracted and added to the value.
keep.index	Logical. If FALSE remove the bootstrap or cross-validation index from the output list which otherwise is included in the splitMethod part of the output list.
keep.matrix	Logical. If TRUE add all B prediction error curves from bootstrapping or cross-validation to the output.
keep.models	Logical. If TRUE keep the models in object. Since fitted models can be large objects the default is FALSE.
keep.residuals	Logical. If TRUE keep the patient individual residuals at testTimes.
keep.pvalues	For multiSplitTest. If TRUE keep the pvalues from the single splits.
noinf.permute	If TRUE the noinformation error is approximated using permutation.
multiSplitTest	If TRUE the test proposed by van de Wiel et al. (2009) is applied. Requires subsampling bootstrap cross-validation, i.e. that splitMethod equals bootcv and that M is specified.
testIBS	A range of time points for testing differences between models in the integrated Brier scores.
testTimes	A vector of time points for testing differences between models in the time-point specific Brier scores.
confInt	Experimental.
confLevel	Experimental.
verbose	if TRUE report details of the progress, e.g. count the steps in cross-validation.
savePath	Place in your file system (i.e., a directory on your computer) where training models fitted during cross-validation are saved. If missing training models are not saved.
slaveseed	Vector of seeds, as long as B, to be given to the slaves in parallel computing.
na.action	Passed immediately to model.frame. Defaults to na.fail. If set otherwise most prediction models will not work.
...	Not used.

Details

Note that package riskRegression provides very similar functionality (and much more) but not yet every feature of pec.

Missing data in the response or in the input matrix cause a failure.

The status of the continuous response variable at cutpoints (times), ie status=1 if the response value exceeds the cutpoint and status=0 otherwise, is compared to predicted event status probabilities

which are provided by the prediction models on the basis of covariates. The comparison is done with the Brier score: the quadratic difference between 0-1 response status and predicted probability.

There are two different sources for bias when estimating prediction error in right censored survival problems: censoring and high flexibility of the prediction model. The first is controlled by inverse probability of censoring weighting, the second can be controlled by special Monte Carlo simulation. In each step, the resampling procedures reevaluate the prediction model. Technically this is done by replacing the argument `object$call$data` by the current subset or bootstrap sample of the full data.

For each prediction model there must be a `predictSurvProb` method: for example, to assess a prediction model which evaluates to a `myclass` object one defines a function called `predictSurvProb.myclass` with arguments `object,newdata,cutpoints,...`

Such a function takes the object and derives a matrix with predicted event status probabilities for each subject in newdata (rows) and each cutpoint (column) of the response variable that defines an event status.

Currently, `predictSurvProb` methods are available for the following R-objects:

`matrix`

`aalen`, `cox.aalen` from `library(timereg)`

`mfp` from `library(mfp)`

`phnnet`, `survnnet` from `library(survnnet)`

`rpart` (from `library(rpart)`)

`coxph`, `survfit` from `library(survival)`

`cph`, `psm` from `library(rms)`

`prodlim` from `library(prodlim)`

`glm` from `library(stats)`

Value

A `pec` object. See also the help pages of the corresponding `print`, `summary`, and `plot` methods. The object includes the following components:

`PredErr`	The estimated prediction error according to the `splitMethod`. A matrix where each column represents the estimated prediction error of a fit at the time points in time.
`AppErr`	The training error or apparent error obtained when the model(s) are evaluated in the same data where they were trained. Only if `splitMethod` is one of "NoInf", "cvK", "BootCv", "Boot632" or "Boot632plus".
`BootCvErr`	The prediction error when the model(s) are trained in the bootstrap sample and evaluated in the data that are not in the bootstrap sample. Only if `splitMethod` is one of "Boot632" or "Boot632plus". When `splitMethod="BootCv"` then the `BootCvErr` is stored in the component `PredErr`.
`NoInfErr`	The prediction error when the model(s) are evaluated in the permuted data. Only if `splitMethod` is one of "BootCv", "Boot632", or "Boot632plus". For `splitMethod="NoInf"` the `NoInfErr` is stored in the component `PredErr`.

weight	The weight used to linear combine the AppErr and the BootCvErr Only if splitMethod is one of "Boot632", or "Boot632plus".
overfit	Estimated overfit of the model(s). See Efron \& Tibshirani (1997, Journal of the American Statistical Association) and Gerds \& Schumacher (2007, Biometrics). Only if splitMethod is one of "Boot632", or "Boot632plus".
call	The call that produced the object
time	The time points at which the prediction error curves change.
ipcw.fit	The fitted censoring model that was used for re-weighting the Brier score residuals. See Gerds \& Schumacher (2006, Biometrical Journal)
n.risk	The number of subjects at risk for all time points.
models	The prediction models fitted in their own data.
cens.model	The censoring models.
maxtime	The latest timepoint where the prediction error curves are estimated.
start	The earliest timepoint where the prediction error curves are estimated.
exact	TRUE if the prediction error curves are estimated at all unique values of the response in the full data.
splitMethod	The splitMethod used for estimation of the overfitting bias.

Author(s)

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

References

Gerds TA, Kattan MW. Medical Risk Prediction Models: With Ties to Machine Learning. Chapman & Hall/CRC https://www.routledge.com/9781138384477

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

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

Efron, Tibshirani (1997) Journal of the American Statistical Association 92, 548–560 Improvement On Cross-Validation: The .632+ Bootstrap Method.

Gerds, Schumacher (2006), Consistent estimation of the expected Brier score in general survival models with right-censored event times. Biometrical Journal, vol 48, 1029–1040.

Thomas A. Gerds, Martin Schumacher (2007) Efron-Type Measures of Prediction Error for Survival Analysis Biometrics, 63(4), 1283–1287 doi:10.1111/j.1541-0420.2007.00832.x

Martin Schumacher, Harald Binder, and Thomas Gerds. Assessment of survival prediction models based on microarray data. Bioinformatics, 23(14):1768-74, 2007.

Mark A. van de Wiel, Johannes Berkhof, and Wessel N. van Wieringen Testing the prediction error difference between 2 predictors Biostatistics (2009) 10(3): 550-560 doi:10.1093/biostatistics/kxp011

See Also

plot.pec, summary.pec, R2, crps

Examples

```
# simulate an artificial data frame
# with survival response and two predictors

set.seed(130971)
library(prodlim)
library(survival)
dat <- SimSurv(100)

# fit some candidate Cox models and compute the Kaplan-Meier estimate

Models <- list("Cox.X1"=coxph(Surv(time,status)~X1,data=dat,x=TRUE,y=TRUE),
               "Cox.X2"=coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE),
               "Cox.X1.X2"=coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE))

# compute the apparent prediction error
PredError <- pec(object=Models,
                 formula=Surv(time,status)~X1+X2,
                 data=dat,
                 exact=TRUE,
                 cens.model="marginal",
                 splitMethod="none",
                 B=0,
                 verbose=TRUE)

print(PredError,times=seq(5,30,5))
summary(PredError)
plot(PredError,xlim=c(0,30))

# Comparison of Weibull model and Cox model
library(survival)
library(rms)
library(pec)
data(pbc)
pbc <- pbc[sample(1:NROW(pbc),size=100),]
f1 <- psm(Surv(time,status!=0)~edema+log(bili)+age+sex+albumin,data=pbc)
f2 <- coxph(Surv(time,status!=0)~edema+log(bili)+age+sex+albumin,data=pbc,x=TRUE,y=TRUE)
f3 <- cph(Surv(time,status!=0)~edema+log(bili)+age+sex+albumin,data=pbc,surv=TRUE)
brier <- pec(list("Weibull"=f1,"CoxPH"=f2,"CPH"=f3),data=pbc,formula=Surv(time,status!=0)~1)
print(brier)
plot(brier)

# compute the .632+ estimate of the generalization error
set.seed(130971)
library(prodlim)
library(survival)
dat <- SimSurv(100)
```

```
set.seed(17100)
PredError.632plus <- pec(object=Models,
                  formula=Surv(time,status)~X1+X2,
                  data=dat,
                  exact=TRUE,
                  cens.model="marginal",
                  splitMethod="Boot632plus",
                  B=3,
                  verbose=TRUE)

print(PredError.632plus,times=seq(4,12,4))
summary(PredError.632plus)
plot(PredError.632plus,xlim=c(0,30))
# do the same again but now in parallel
## Not run: set.seed(17100)
# library(doMC)
# registerDoMC()
PredError.632plus <- pec(object=Models,
                  formula=Surv(time,status)~X1+X2,
                  data=dat,
                  exact=TRUE,
                  cens.model="marginal",
                  splitMethod="Boot632plus",
                  B=3,
                  verbose=TRUE)

## End(Not run)
# assessing parametric survival models in learn/validation setting
learndat <- SimSurv(50)
testdat <- SimSurv(30)
library(rms)
f1 <- psm(Surv(time,status)~X1+X2,data=learndat)
f2 <- psm(Surv(time,status)~X1,data=learndat)
pf <- pec(list(f1,f2),formula=Surv(time,status)~1,data=testdat,maxtime=200)
plot(pf)
summary(pf)

# ---------------- competing risks -----------------

library(survival)
library(riskRegression)
if(requireNamespace("cmprsk",quietly=TRUE)){
library(cmprsk)
library(pec)
cdat <- SimCompRisk(100)
f1  <- CSC(Hist(time,event)~X1+X2,cause=2,data=cdat)
f2  <- CSC(Hist(time,event)~X1,data=cdat,cause=2)
f3  <- FGR(Hist(time,event)~X1+X2,cause=2,data=cdat)
f4  <- FGR(Hist(time,event)~X1+X2,cause=2,data=cdat)
p1 <- pec(list(f1,f2,f3,f4),formula=Hist(time,event)~1,data=cdat,cause=2)
}
```

pecCforest *S3-wrapper function for cforest from the party package*

Description

S3-wrapper function for cforest from the party package

Usage

```
pecCforest(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

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

pecCtree *S3-Wrapper for ctree.*

Description

The call is added to an ctree object

Usage

```
pecCtree(...)
```

Arguments

...	passed to ctree

Value

list with two elements: ctree and call

Author(s)

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

See Also

pecCforest

Examples

```
if (requireNamespace("party",quietly=TRUE)){
library(prodlim)
library(survival)
set.seed(50)
d <- SimSurv(50)
nd <- data.frame(X1=c(0,1,0),X2=c(-1,0,1))
f <- pecCtree(Surv(time,status)~X1+X2,data=d)
predictSurvProb(f,newdata=nd,times=c(3,8))
}
```

---

pecRpart *Predict survival based on rpart tree object*

Description

Combines the rpart result with a stratified Kaplan-Meier (prodlim) to predict survival

Usage

```
pecRpart(formula, data, ...)
```

Arguments

formula	passed to rpart
data	passed to rpart
...	passed to rpart

Value

list with three elements: ctree and call

Examples

```
library(prodlim)
if (!requireNamespace("rpart",quietly=TRUE)){
library(rpart)
library(survival)
set.seed(50)
d <- SimSurv(50)
nd <- data.frame(X1=c(0,1,0),X2=c(-1,0,1))
f <- pecRpart(Surv(time,status)~X1+X2,data=d)
predictSurvProb(f,newdata=nd,times=c(3,8))
}
```

---

plot.calibrationPlot *Plot objects obtained with* calPlot

Description

Calibration plots

Usage

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

Arguments

x	Object obtained with calPlot
...	Not used.

Value

Nothing

Author(s)

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

See Also

calPlot

plot.pec *Plotting prediction error curves*

Description

Plotting prediction error curves for one or more prediction models.

Usage

```
## S3 method for class 'pec'
plot(
  x,
  what,
  models,
  xlim = c(x$start, x$minmaxtime),
  ylim = c(0, 0.3),
  xlab = "Time",
  ylab,
  axes = TRUE,
  col,
  lty,
  lwd,
  type,
  smooth = FALSE,
  add.refline = FALSE,
  add = FALSE,
  legend = ifelse(add, FALSE, TRUE),
  special = FALSE,
  ...
)
```

Arguments

x	Object of class pec obtained with function pec.
what	The name of the entry in x. Defauls to PredErr Other choices are AppErr, BootCvErr, Boot632, Boot632plus.
models	Specifies models in x$models for which the prediction error curves are drawn. Defaults to all models.
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 curves of models in the order determined by models.

lty	Vector of lty's given to the curves of models in the order determined by models.
lwd	Vector of lwd's given to the curves of models in the order determined by models.
type	Plotting type: either "l" or "s", see lines.
smooth	Logical. If TRUE the plotting type for lines is 'l' else 's'.
add.refline	Logical. If TRUE a dotted horizontal line is drawn as a symbol for the naive rule that predicts probability .5 at all cutpoints (i.e. time points in survival analysis).
add	Logical. If TRUE only lines are added to an existing device
legend	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.
special	Logical. If TRUE the bootstrap curves of models are plotted together with predErr of models by invoking the function Special. Optional arguments of the function Special can be given in the form special.x=val as with legend. See also Details.
...	Extra arguments that are passed to plot.

Details

From version 2.0.1 on the arguments legend.text, legend.args, lines.type, lwd.lines, specials are obsolete and only available for backward compatibility. Instead arguments for the invoked functions legend, axis, Special are simply 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 plot.pec and inside by using the function resolveSmartArgs. Documentation of these arguments can be found in the help pages of the corresponding functions.

Value

The (invisible) object.

Author(s)

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

See Also

pecsummary.pecSpecialprodlim

Examples

```
# simulate data
# with a survival response and two predictors
library(prodlim)
library(survival)
```

```
set.seed(280180)
dat <- SimSurv(100)

# fit some candidate Cox models and
# compute the Kaplan-Meier estimate

Models <- list("Kaplan.Meier"=survfit(Surv(time,status)~1,data=dat),
               "Cox.X1"=coxph(Surv(time,status)~X1,data=dat,x=TRUE,y=TRUE),
               "Cox.X2"=coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE),
               "Cox.X1.X2"=coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE))
Models <- list("Cox.X1"=coxph(Surv(time,status)~X1,data=dat,x=TRUE,y=TRUE),
               "Cox.X2"=coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE),
               "Cox.X1.X2"=coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE))


# compute the .632+ estimate of the generalization error
set.seed(17100)
PredError.632plus <- pec(object=Models,
                         formula=Surv(time,status)~X1+X2,
                         data=dat,
                         exact=TRUE,
                         cens.model="marginal",
                         splitMethod="boot632plus",
                         B=5,
                         keep.matrix=TRUE,
                         verbose=TRUE)

# plot the .632+ estimates of the generalization error
plot(PredError.632plus,xlim=c(0,30))

# plot the bootstrapped curves, .632+ estimates of the generalization error
# and Apparent error for the Cox model 'Cox.X1' with the 'Cox.X2' model
# as benchmark
plot(PredError.632plus,
     xlim=c(0,30),
     models="Cox.X1",
     special=TRUE,
     special.bench="Cox.X2",
     special.benchcol=2,
     special.addprederr="AppErr")
```

---

plotPredictEventProb *Plotting predicted survival curves.*

Description

Ploting time-dependent event risk predictions.

Usage

```
plotPredictEventProb(
  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	Densitiy 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` are simply 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 `plotPredictEventProb` and inside by using the function `SmartControl`.

Value

The (invisible) object.

Author(s)

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

References

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

See Also

`predictEventProbprodlim`

Examples

```
# generate some competing risk data
```

`plotPredictSurvProb` *Plotting predicted survival curves.*

Description

Ploting prediction survival curves for one prediction model using `predictSurvProb` .

Usage

```
plotPredictSurvProb(
  x,
  newdata,
  times,
  xlim,
  ylim,
```

```
  xlab,
  ylab,
  axes = TRUE,
  col,
  density,
  lty,
  lwd,
  add = FALSE,
  legend = TRUE,
  percent = FALSE,
  ...
)
```

Arguments

x	A survival prediction model including call and formula object.
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.
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	Densitiy 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 are simply 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 plotPredictSurvProb and inside by using the function SmartControl.

Value

The (invisible) object.

Author(s)

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

References

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

See Also

predictSurvProbprodlim

Examples

```
# generate some survival data
library(prodlim)
d <- SimSurv(100)
# then fit a Cox model
library(rms)
coxmodel <- cph(Surv(time,status)~X1+X2,data=d,surv=TRUE)
# plot predicted survival probabilities for all time points
ttt <- sort(unique(d$time))
# and for selected predictor values:
 ndat <- data.frame(X1=c(0.25,0.25,-0.05,0.05),X2=c(0,1,0,1))
plotPredictSurvProb(coxmodel,newdata=ndat,times=ttt)

# the same can be done e.g. for a randomSurvivalForest model
library(randomForestSRC)
rsfmodel <- rfsrc(Surv(time,status)~X1+X2,data=d)
plotPredictSurvProb(rsfmodel,newdata=ndat,times=ttt)
```

---

predictEventProb *Predicting event probabilities (cumulative incidences) in competing risk models.*

Description

Function to extract event probability predictions from various modeling approaches. The most prominent one is the combination of cause-specific Cox regression models which can be fitted with the function cumincCox from the package compRisk.

Usage

```
predictEventProb(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.
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.
...	Additional arguments that are passed on to the current method.

Details

The function predictEventProb is a generic function that means it invokes specifically designed functions depending on the 'class' of the first argument.

See predictSurvProb.

Value

A matrix with as many rows as NROW(newdata) and as many columns as length(times). Each entry should be a probability and in rows the values should be increasing.

Author(s)

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

See Also

See predictSurvProb.

Examples

```
library(pec)
library(survival)
library(riskRegression)
library(prodlim)
train <- SimCompRisk(100)
test <- SimCompRisk(10)
cox.fit  <- CSC(Hist(time,cause)~X1+X2,data=train)
predictEventProb(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)
predictEventProb(cox.fit2,newdata=test,times=seq(1:10),cause=1)
```

predictLifeYearsLost *Predicting life years lost (cumulative cumulative incidences) in competing risk models.*

Description

Function to extract predicted life years lost from various modeling approaches. The most prominent one is the combination of cause-specific Cox regression models which can be fitted with the function `cumincCox` from the package `compRisk`.

Usage

```
predictLifeYearsLost(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.
`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.
`...`	Additional arguments that are passed on to the current method.

Details

The function predictLifeYearsLost is a generic function that means it invokes specifically designed functions depending on the 'class' of the first argument.

See `predictSurvProb`.

Value

A matrix with as many rows as `NROW(newdata)` and as many columns as `length(times)`. Each entry should be a positive value and in rows the values should be increasing.

Author(s)

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

See Also

`predictSurvProb`, `predictEventProb`.

Examples

```
library(pec)
library(riskRegression)
library(survival)
library(prodlim)
train <- SimCompRisk(100)
test <- SimCompRisk(10)
fit <- CSC(Hist(time,cause)~X1+X2,data=train,cause=1)
predictLifeYearsLost(fit,newdata=test,times=seq(1:10),cv=FALSE,cause=1)
```

---

predictRestrictedMeanTime

Predicting restricted mean time

Description

Function to extract predicted mean times from various modeling approaches.

Usage

```
## S3 method for class 'aalen'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'riskRegression'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'cox.aalen'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'cph'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'coxph'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'matrix'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'selectCox'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'prodlim'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'psm'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'survfit'
predictRestrictedMeanTime(object,newdata,times,...)
## S3 method for class 'pecRpart'
predictRestrictedMeanTime(object,newdata,times,...)
#' \method{predictRestrictedMeanTime}{pecCtree}(object,newdata,times,...)
```

Arguments

object	A fitted model from which to extract predicted survival probabilities
newdata	A data frame containing predictor variable combinations for which to compute predicted survival probabilities.
times	A vector of times in the range of the response variable, e.g. times when the response is a survival object, at which to return the survival probabilities.
...	Additional arguments that are passed on to the current method.

Details

The function predictRestrictedMeanTime is a generic function, meaning that it invokes a different function dependent on the 'class' of the first argument.

See also predictSurvProb.

Value

A matrix with as many rows as NROW(newdata) and as many columns as length(times). Each entry should be a probability and in rows the values should be decreasing.

Note

In order to assess the predictive performance of a new survival model a specific predictRestrictedMeanTime S3 method has to be written. For examples, see the bodies of the existing methods.

The performance of the assessment procedure, in particular for resampling where the model is repeatedly evaluated, will be improved by supressing in the call to the model all the computations that are not needed for probability prediction. For example, se.fit=FALSE can be set in the call to cph.

Author(s)

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

References

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

See Also

predict,survfit

Examples

```
# generate some survival data
library(prodlim)
set.seed(100)
```

```
d <- SimSurv(100)
# then fit a Cox model
library(rms)
coxmodel <- cph(Surv(time,status)~X1+X2,data=d,surv=TRUE)

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

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

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

# the same can be done e.g. for a randomSurvivalForest model
library(randomForestSRC)
rsfmodel <- rfsrc(Surv(time,status)~X1+X2,data=d)
predictRestrictedMeanTime(rsfmodel,newdata=ndat,times=ttt)
```

`predictSurvProb` *Predicting survival probabilities*

Description

Function to extract survival probability predictions from various modeling approaches. The most prominent one is the Cox regression model which can be fitted for example with 'coxph' and with 'cph'.

Usage

```
## S3 method for class 'aalen'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'riskRegression'
```

```
predictSurvProb(object,newdata,times,...)
## S3 method for class 'cox.aalen'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'cph'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'coxph'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'matrix'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'selectCox'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'pecCforest'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'prodlim'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'psm'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'survfit'
predictSurvProb(object,newdata,times,...)
## S3 method for class 'pecRpart'
predictSurvProb(object,newdata,times,...)
#' \method{predictSurvProb}{pecCtree}(object,newdata,times,...)
```

Arguments

`object`	A fitted model from which to extract predicted survival probabilities
`newdata`	A data frame containing predictor variable combinations for which to compute predicted survival probabilities.
`times`	A vector of times in the range of the response variable, e.g. times when the response is a survival object, at which to return the survival probabilities.
`...`	Additional arguments that are passed on to the current method.

Details

The function predictSurvProb is a generic function that means it invokes specifically designed functions depending on the 'class' of the first argument.

The function `pec` requires survival probabilities for each row in newdata at requested times. These probabilities are extracted from a fitted model of class `CLASS` with the function `predictSurvProb.CLASS`.

Currently there are `predictSurvProb` methods for objects of class cph (library rms), coxph (library survival), aalen (library timereg), cox.aalen (library timereg), rpart (library rpart), product.limit (library prodlim), survfit (library survival), psm (library rms)

Value

A matrix with as many rows as `NROW(newdata)` and as many columns as `length(times)`. Each entry should be a probability and in rows the values should be decreasing.

Note

In order to assess the predictive performance of a new survival model a specific `predictSurvProb` S3 method has to be written. For examples, see the bodies of the existing methods.

The performance of the assessment procedure, in particular for resampling where the model is repeatedly evaluated, will be improved by supressing in the call to the model all the computations that are not needed for probability prediction. For example, `se.fit=FALSE` can be set in the call to `cph`.

Author(s)

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

References

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

See Also

predict,survfit

Examples

```
# generate some survival data
library(prodlim)
set.seed(100)
d <- SimSurv(100)
# then fit a Cox model
library(rms)
coxmodel <- cph(Surv(time,status)~X1+X2,data=d,surv=TRUE)

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

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

## simulate some learning and some validation data
learndat <- SimSurv(100)
valdat <- SimSurv(100)
## use the learning data to fit a Cox model
library(survival)
fitCox <- coxph(Surv(time,status)~X1+X2,data=learndat,x=TRUE,y=TRUE)
```

```
## suppose we want to predict the survival probabilities for all patients
## in the validation data at the following time points:
## 0, 12, 24, 36, 48, 60
psurv <- predictSurvProb(fitCox,newdata=valdat,times=seq(0,60,12))
## This is a matrix with survival probabilities
## 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,status)~X1+X2,data=d)
predictSurvProb(rsfmodel,newdata=ndat,times=ttt)

## Cox with ridge option
f1 <- coxph(Surv(time,status)~X1+X2,data=learndat,x=TRUE,y=TRUE)
f2 <- coxph(Surv(time,status)~ridge(X1)+ridge(X2),data=learndat,x=TRUE,y=TRUE)
plot(predictSurvProb(f1,newdata=valdat,times=10),
     pec:::predictSurvProb.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")
```

`print.pec` *Printing a 'pec' (prediction error curve) object.*

Description

Print the important arguments of call and the prediction error values at selected time points.

Usage

```
## S3 method for class 'pec'
print(x, times, digits = 3, what = NULL, ...)
```

Arguments

`x`	Object of class pec
`times`	Time points at which to show the values of the prediction error curve(s)
`digits`	Number of decimals used in tables.
`what`	What estimate of the prediction error curve to show. Should be a string matching an element of x. The default is determined by splitMethod.
`...`	Not used
`print`	Set to FALSE to suppress printing.

Value

The first argument in the invisible cloak.

Author(s)

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

See Also

pec

R2 *Explained variation for survival models*

Description

This function computes a time-dependent R^2 like measure of the variation explained by a survival prediction model, by dividing the mean squared error (Brier score) of the model by the mean squared error (Brier score) of a reference model which ignores all the covariates.

Usage

```
R2(object, models, what, times, reference = 1)
```

Arguments

object	An object with estimated prediction error curves obtained with the function pec
models	For which of the models in object$models should we compute $R^2(t). By default all models are used except for the reference model.
what	The name of the entry in x to be used. Defauls to PredErr Other choices are AppErr, BootCvErr, Boot632, Boot632plus.
times	Time points at which the summaries are shown.
reference	Position of the model whose prediction error is used as the reference in the denominator when constructing R^2

Details

In survival analysis the prediction error of the Kaplan-Meier estimator plays a similar role as the total sum of squares in linear regression. Hence, it is a sensible reference model for R^2.

Value

A matrix where the first column holds the times and the following columns are the corresponding R^2 values for the requested prediction models.

Author(s)

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

References

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

Gerds TA, Cai T & Schumacher M (2008) The performance of risk prediction models Biometrical Journal, 50(4), 457–479

See Also

pec

Examples

```
set.seed(18713)
library(prodlim)
library(survival)
dat=SimSurv(100)
nullmodel=prodlim(Hist(time,status)~1,data=dat)
pmodel1=coxph(Surv(time,status)~X1+X2,data=dat,x=TRUE,y=TRUE)
pmodel2=coxph(Surv(time,status)~X2,data=dat,x=TRUE,y=TRUE)
perror=pec(list(Cox1=pmodel1,Cox2=pmodel2),Hist(time,status)~1,data=dat,reference=TRUE)
R2(perror,times=seq(0,1,.1),reference=1)
```

---

reclass *Retrospective risk reclassification table*

Description

Retrospective table of risks predicted by two different methods, models, algorithms

Usage

```
reclass(
  object,
  reference,
  formula,
  data,
  time,
  cause,
  cuts = seq(0, 100, 25),
  digits = 2
)
```

Arguments

object	Either a list with two elements. Each element should either be a vector with probabilities, or an object for which predictSurvProb or predictEventProb can extract predicted risk based on data.
reference	Reference prediction model.
formula	A survival formula as obtained either with prodlim::Hist or survival::Surv which defines the response in the data.
data	Used to extract the response from the data and passed on to predictEventProb to extract predicted event probabilities.
time	Time interest for prediction.
cause	For competing risk models the cause of interest. Defaults to all available causes.
cuts	Risk quantiles to group risks.
digits	Number of digits to show for the predicted risks

Details

All risks are multiplied by 100 before

Value

reclassification tables: overall table and one conditional table for each cause and for subjects event free at time interest.

Author(s)

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

See Also

predictStatusProb

Examples

```
## Not run:
library(survival)
set.seed(40)
d <- prodlim::SimSurv(400)
nd <- prodlim::SimSurv(400)
Models <- list("Cox.X2"=coxph(Surv(time,status)~X2,data=d,x=TRUE,y=TRUE),
               "Cox.X1.X2"=coxph(Surv(time,status)~X1+X2,data=d,x=TRUE,y=TRUE))
rc <- reclass(Models,formula=Surv(time,status)~1,data=nd,time=5)
print(rc)
plot(rc)

set.seed(40)
library(riskRegression)
library(prodlim)
dcr <- prodlim::SimCompRisk(400)
```

```
ndcr <- prodlim::SimCompRisk(400)
crPred5 <- list("X2"=predictEventProb(CSC(Hist(time,event)~X2,data=dcr),newdata=ndcr,times=5),
          "X1+X2"=predictEventProb(CSC(Hist(time,event)~X1+X2,data=dcr),newdata=ndcr,times=5))
rc <- reclass(crPred5,Hist(time,event)~1,data=ndcr,time=3)
print(rc)

reclass(crPred5,Hist(time,event)~1,data=ndcr,time=5,cuts=100*c(0,0.05,0.1,0.2,1))

## End(Not run)
```

`resolvesplitMethod` *Resolve the splitMethod for estimation of prediction performance*

Description

The function computes a matrix of random indices obtained by drawing from the row numbers of a data set either with or without replacement. The matrix can be used to repeatedly set up independent training and validation sets.

Usage

```
resolvesplitMethod(splitMethod, B, N, M)
```

Arguments

`splitMethod`	String that determines the splitMethod to use. Available splitMethods are none/noPlan (no splitting), bootcv or outofbag (bootstrap cross-validation), cvK (K-fold cross-validation, e.g. cv10 gives 10-fold), boot632, boot632plus or boot632+, loocv (leave-one-out)
`B`	The number of repetitions.
`N`	The sample size
`M`	For subsampling bootstrap the size of the subsample. Note M<N.

Value

A list with the following components

`name`	the official name of the splitMethod
`internal.name`	the internal name of the splitMethod
`index`	a matrix of indices with B columns and either N or M rows, dependent on splitMethod
`B`	the value of the argument B
`N`	the value of the argument N
`M`	the value of the argument M

Author(s)

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

Examples

```
# BootstrapCrossValidation: Sampling with replacement
resolvesplitMethod("BootCv",N=10,B=10)

# 10-fold cross-validation: repeated 2 times
resolvesplitMethod("cv10",N=10,B=2)

# leave-one-out cross-validation
resolvesplitMethod("loocv",N=10)

resolvesplitMethod("bootcv632plus",N=10,B=2)
```

---

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

```
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

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. Journal of Statistical Software, 50(11), 1-23. DOI 10.18637/jss.v050.i11

Examples

```
data(GBSG2)
library(survival)
f <- selectCox(Surv(time,cens)~horTh+age+menostat+tsize+tgrade+pnodes+progrec+estrec ,
       data=GBSG2)
```

`selectFGR` *Stepwise variable selection in the Fine & Gray regression competing risk model*

Description

This is a wrapper function which first selects variables in the Fine & Gray regression model using `crrstep` from the `crrstep` package and then returns a fitted Fine & Gray regression model with the selected variables.

Usage

```
selectFGR(formula, data, cause = 1, rule = "AIC", direction = "backward", ...)
```

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`.
`rule`	Rule to pass on to crrstep ("AIC", "BIC" or "BICcr"), also see `crrstep`
`direction`	see `crrstep`
`...`	Further arguments passed to `crrstep`.

Author(s)

Rob C.M. van Kruijsdijk <R.C.M.vanKruijsdijk@umcutrecht.nl>

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

Examples

```
## Not run:
library(riskRegression)
library(prodlim)
library(lava)
if (!requireNamespace("cmprsk",quietly=TRUE)){
library(cmprsk)
library(pec)
```

```
m <- crModel()
m <- addvar(m,c('X1','X2','X3','X4','X5','X6','X7','X8','X9','X10'))
distribution(m,c("X2","X7","X9")) <- binomial.lvm()
regression(m,eventtime1~X1+X2+X5+X9) <- c(-1,1,0.5,0.8)
set.seed(100)
d <- sim(m,100)
## full formula
ff <- Hist(time, event) ~ X1 + X2 + X3 + X4 +X5 + X6 + X7+ X8 + X9 + X10

# Fit full model with FGR
fg <- FGR(ff,cause=1,data=d)

# Backward selection based on the AIC
sfgAIC <- selectFGR(ff, data=d, rule="AIC", direction="backward")

sfgAIC$fit # Final FGR-model with selected variables

# Risk reclassification plot at time = 4
plot(predictEventProb(fg,times=4,newdata=d),
     predictEventProb(sfgAIC,times=4,newdata=d))

# Backward selection based on the BIC, while forcing
# the last two variables (X9 and X10) in the model
sfgBIC <- selectFGR(ff, data=d, rule="BIC", direction="backward",
                  scope.min=~X9+X10)

## apparent performance
pec(list(full.model=fg,selectedAIC=sfgAIC,selectedBIC=sfgBIC),
    formula=Hist(time, event)~1,
    data=d)


## bootstrap cross-validation performance
set.seed(7)
pec(list(full.model=fg,selectedAIC=sfgAIC,selectedBIC=sfgBIC),
    formula=Hist(time, event)~1,
    data=d,
    B=5,
    splitMethod="bootcv")
}

## End(Not run)
```

---

simCost *Simulate COST alike data*

Description

Simulate data alike the data from the Copenhagen stroke study (COST)

Usage

```
simCost(N)
```

Arguments

N	Sample size

Details

This uses functionality of the lava package.

Value

Data frame

Author(s)

Thomas Alexander Gerds

Special *Drawing bootstrapped cross-validation curves and the .632 or .632plus error of models. The prediction error for an optional benchmark model can be added together with bootstrapped cross-validation error and apparent errors.*

Description

This function is invoked and controlled by `plot.pec`.

Usage

```
Special(
  x,
  y,
  addprederr,
  models,
  bench,
  benchcol,
  times,
  maxboot,
  bootcol,
  col,
  lty,
  lwd
)
```

Arguments

x	an object of class 'pec' as returned by the pec function.
y	Prediction error values.
addprederr	Additional prediction errors. The options are bootstrap cross-validation errors or apparent errors.
models	One model also specified in pec for which the predErr in plot.pec is to be drawn.
bench	A benchmark model (also specified in pec) for which the predErr in plot.pec is to be drawn.
benchcol	Color of the benchmark curve.
times	Time points at which the curves must be plotted.
maxboot	Maximum number of bootstrap curves to be added. Default is all.
bootcol	Color of the bootstrapped curves. Default is 'gray77'.
col	Color of the different error curves for models.
lty	Line type of the different error curves for models.
lwd	Line width of the different error curves for models.

Details

This function should not be called directly. The arguments can be specified as Special.arg in the call to plot.pec.

Value

Invisible object.

See Also

plot.pec

---

threecity *threecity data*

Description

Extracted data from a french population based cohort (Three-City cohort). The dataset includes followup information on dementia outcome and predicted 5-year risks based on based on the subject specific information which includes age, gender, education level and cognitive decline measured by a psychometric test (Mini Mental State Examination). The prediction model from which the predictions have been computed has been fitted on independent training data from the Paquid cohort, another french population based cohort with similar design (see Reference Blanche et al. 2015 for details) .

Format

A subsample consisting of 2000 observations on the following 3 variables.

pi 5-year absolute risk predictions of dementia.

status 0=censored, 1=dementia, 2=death dementia free

time time to event (i.e., time to either dementia, death dementia free or loss of follow-up)

Source

Web-appendix of Blanche et al. (2015).

References

Blanche, P., Proust-Lima, C., Loubere, L., Berr, C., Dartigues, J. F., Jacqmin-Gadda, H. (2015). Quantifying and comparing dynamic predictive accuracy of joint models for longitudinal marker and time-to-event in presence of censoring and competing risks. Biometrics, 71(1), 102-113.

Examples

```
data(threecity)
```

Index