

# Package ‘pec’

February 15, 2012

**Title** Prediction Error Curves for Survival Models

**Version** 2.1.7

**Author** Thomas A. Gerds

**Description** Validation of predicted survival probabilities using inverse weighting and resampling.

**Depends** R (>= 1.9.1), prodlim (>= 1.1.3), survival, rms

**Suggests** timereg, randomSurvivalForest, party, rpart, cmprsk, foreach, doMC

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

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2012-01-30 20:18:19

## R topics documented:

cindex . . . . .	2
crps . . . . .	6
GBSG2 . . . . .	8
ipcw . . . . .	9
pec . . . . .	11
pecCforest . . . . .	18
plot.pec . . . . .	18
plotPredictEventProb . . . . .	21
plotPredictSurvProb . . . . .	23
predictEventProb . . . . .	25
predictSurvProb . . . . .	26
print.pec . . . . .	28
R2 . . . . .	29
resolvesplitMethod . . . . .	30
selectCox . . . . .	31
Special . . . . .	32
<b>Index</b>	<b>34</b>

---

`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 weighted 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,...)
## S3 method for class 'list'
cindex(object,
        formula,
        data,
        eval.times,
        pred.times,
        cause,
        cens.model="marginal",
        ipcw.refit=FALSE,
        ipcw.limit,
        tiedPredictionsIn=TRUE,
        tiedOutcomeIn=TRUE,
        tiedMatchIn=TRUE,
        splitMethod="noPlan",
        B,
        M,
        model.args=NULL,
        model.parms=NULL,
        keep.models="Call",
        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,
```

...)

**Arguments**

<code>object</code>	A named list of prediction models, where allowed entries are (1) R-objects for which a <code>predictSurvProb</code> 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.
<code>formula</code>	A survival formula. 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 <code>Surv(time, status)~x1+x2</code> and <code>cens.model="cox"</code> . Then the weights are based on a Cox regression model for the censoring times with predictors <code>x1</code> and <code>x2</code> . Note that the usual coding is assumed: <code>status=0</code> for censored times and that each variable name that appears in <code>formula</code> must be the column name in data. If there are no covariates, i.e. <code>formula=Surv(time,status)~1</code> the <code>cens.model</code> is coerced to "marginal" and the Kaplan-Meier estimator for the censoring times is used to calculate the weights. If <code>formula</code> is missing the formula of the first model in list is used.
<code>data</code>	A data frame in which to validate the prediction models and to fit the censoring model. If <code>data</code> is missing, the data of the first model in list is used.
<code>eval.times</code>	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 <code>eval.times</code> is missing or <code>Inf</code> then the largest uncensored event time is used.
<code>pred.times</code>	A vector of timepoints for evaluating the prediction models. This should either be exactly one timepoint used for all <code>eval.times</code> , or be as long as <code>eval.times</code> , in which case the predicted order of risk for the <code>j</code> th entry of <code>eval.times</code> is based on the <code>j</code> th entry of <code>pred.times</code> corresponding
<code>cause</code>	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 <code>formula</code> in <code>data</code> .
<code>cens.model</code>	Method for estimating inverse probability of censoring weights: <code>cox</code> : A semi-parametric Cox proportional hazard model is fitted to the censoring times <code>marginal</code> : The Kaplan-Meier estimator for the censoring times <code>nonpar</code> : 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 <code>formula</code> but at most one continuous variable. See the documentation of the functions <code>prodlim</code> and <code>neighborhood</code> from the <code>prodlim</code> package. <code>aalen</code> : The nonparametric Aalen additive model fitted to the censoring times. Requires the <code>timereg</code> package maintained by Thomas Scheike.
<code>ipcw.refit</code>	If <code>TRUE</code> the inverse probability of censoring weights are estimated separately in each training set during cross-validation.
<code>ipcw.limit</code>	Value between 0 and 1 (but not equal to 0!) used to cut for small weights in order to stabilize the estimate at late times where few individuals are observed.

<code>tiedPredictionsIn</code>	If FALSE pairs with identical predictions are excluded, unless also the event times are identical and uncensored and <code>tiedMatchIn</code> is set to TRUE.
<code>tiedOutcomeIn</code>	If TRUE pairs with identical and uncensored event times are excluded, unless also the predictions are identical and <code>tiedMatchIn</code> is set to TRUE.
<code>tiedMatchIn</code>	If TRUE then pairs with identical predictions and identical and uncensored event times are counted as concordant pairs.
<code>splitMethod</code>	SplitMethod for estimating the prediction error curves. 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.
<code>B</code>	Number of bootstrap samples. The default depends on argument <code>splitMethod</code> . When <code>splitMethod</code> in <code>c("BootCv","Boot632")</code> the default is 100. For <code>splitMethod="none"</code> B is the number of bootstrap simulations e.g. to obtain bootstrap confidence limits – default is 0.
<code>M</code>	The size of the bootstrap samples for resampling without replacement. Ignored for resampling with replacement.
<code>model.args</code>	List of extra arguments that can be passed to the <code>predictSurvProb</code> methods. The list must have an entry for each entry in object.
<code>model.parms</code>	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.
<code>keep.index</code>	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.
<code>keep.matrix</code>	Logical. If TRUE add all B prediction error curves from bootstrapping or cross-validation to the output.
<code>keep.models</code>	Logical. If TRUE keep the models in object. If "Call" keep only the call of these models.
<code>keep.residuals</code>	Experimental.
<code>keep.pvalues</code>	Experimental.
<code>keep.weights</code>	Experimental.
<code>multiSplitTest</code>	Experimental.
<code>testTimes</code>	A vector of time points for testing differences between models in the time-point specific Brier scores.
<code>confInt</code>	Experimental.
<code>confLevel</code>	Experimental.
<code>verbose</code>	if TRUE report details of the progress, e.g. count the steps in cross-validation.
<code>savePath</code>	Place in your filesystem (directory) where training models fitted during cross-validation are saved. If missing training models are not saved.
<code>...</code>	Not used.

**Details**

Pairs with identical observed times, where one is uncensored and one is censored, are always considered usable (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**

Gerds, TA and Kattan, M and Schumacher, M and Yu, C (2010) Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring Research report 10/7. Department of Biostatistics, University of Copenhagen

**Examples**

```
## Not run:
# simulate data based on Weibull regression
set.seed(13)
dat <- SimSurv(300)
# fit three different Cox models and a random survival forest
# note: low number of trees for the purpose of illustration
library(survival)
library(randomSurvivalForest)
#rsf.default=randomSurvivalForest:::rsf.default
cox12 <- coxph(Surv(time,status)~X1+X2,data=dat)
cox1 <- coxph(Surv(time,status)~X1,data=dat)
cox2 <- coxph(Surv(time,status)~X2,data=dat)
rsf1 <- rsf(Survrsf(time,status)~X1+X2,data=dat,ntree=15,forest=TRUE)
#
# compute the apparent estimate of the C-index at different time points
#
ApparrentCindex <- 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(5,500,50))
print(ApparrentCindex)
plot(ApparrentCindex)
#
# compute the bootstrap-crossvalidation estimate of the C-index at different time points
#
```

```

set.seed(142)
bcvCindex <- 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=10,
                  eval.times=seq(5,500,50))
print(bcvCindex)
plot(bcvCindex)

## End(Not run)
## Not run:
# for uncensored data the results are the same as obtained with the function rcorr.cens from Hmisc
library(Hmisc)
set.seed(16)
dat <- SimSurv(30,cens=FALSE)
fit12 <- coxph(Surv(time,status)~X1+X2,data=dat)
fit1 <- coxph(Surv(time,status)~X1,data=dat)
fit2 <- coxph(Surv(time,status)~X2,data=dat)
Cpec <- cindex(list("Cox X1+X2"=fit12,"Cox X1"=fit1,"Cox X2"=fit2),formula=Surv(time,status)~1,data=dat,eval.times=seq(5,500,50))
p1 <- predictSurvProb(fit1,newdata=dat,times=100)
p2 <- predictSurvProb(fit2,newdata=dat,times=100)
p12 <- predictSurvProb(fit12,newdata=dat,times=100)
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"]]

## End(Not run)

```

---

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)
ibs(object, models, what, times, start)

```

**Arguments**

object	An object with estimated prediction error curves obtained with the function <a href="#">pec</a>
models	Which models in <code>object\$models</code> should be considered.
what	The name of the entry in <code>x</code> to be cumulated. Defaults to <code>PredErr</code> . Other choices are <code>AppErr</code> , <code>BootCvErr</code> , <code>Boot632</code> , <code>Boot632plus</code> .
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](mailto: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)

dat=prodlm::SimSurv(100)
nullmodel=prodlm(Hist(time,status)~1,data=dat)
pmodel=coxph(Surv(time,status)~X1+X2,data=dat)
perror=pec(list(KaplanMeier=nullmodel,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.

### Usage

```
data(GBSG2)
```

### 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).

### Source

<http://www.blackwellpublishers.com/rss/Volumes/A162p1.htm>

### 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  $2 \times 2$  trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*, **12**, 2086–2093.

**Description**

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

**Usage**

```
ipcw(formula,
      data,
      method = c("cox", "marginal", "nonpar", "aalen", "none"),
      times,
      subjectTimes,
      subjectTimesLag=1,
      what)
```

**Arguments**

formula	A survival formula like, $\text{Surv}(\text{time}, \text{status}) \sim 1$ , where as usual $\text{status}=0$ 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.
times	For $\text{what}=\text{"IPCW.times"}$ a vector of times at which to compute the probabilities of not being censored.
subjectTimes	For $\text{what}=\text{"IPCW.subjectTimes"}$ 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.

**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**

<code>times</code>	The times at which weights are estimated
<code>IPCW.times</code>	Estimated weights at <code>times</code>
<code>IPCW.subjectTimes</code>	Estimated weights at individual time values <code>subjectTimes</code>
<code>fit</code>	The fitted censoring model
<code>method</code>	The method for modelling the censoring distribution
<code>call</code>	The call

**Author(s)**

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

**See Also**

[pec](#)

**Examples**

```

dat=prodlm:::SimSurv(300)

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

WCox=ipcw(Surv(time,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(Surv(time,status)~X2,data=dat,method="nonpar",times=sort(unique(dat$time)),subjectTimes=dat$time)
plot(WKM2$fit,add=FALSE)

```

---

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,...)
## S3 method for class 'list'
pec(object,
      formula,
      data,
      times,
      cause,
      start,
      maxtime,
      exact=TRUE,
      exactness=100,
      fillChar=NA,
      cens.model="cox",
      ipcw.refit=FALSE,
      splitMethod="none",
      B,
      M,
      reference=TRUE,
      model.args=NULL,
      model.parms=NULL,
      keep.index=FALSE,
      keep.matrix=FALSE,
      keep.models="Call",
      keep.residuals=FALSE,
      keep.pvalues=FALSE,
      noinf.permute=FALSE,
      multiSplitTest=FALSE,
      testIBS,
      testTimes,
      confInt=FALSE,
      confLevel=0.95,
      verbose=TRUE,
```

```

        savePath=NULL,
        ...)

## S3 method for class 'coxph'
pec(object,...)
## S3 method for class 'glm'
pec(object,...)
## S3 method for class 'cph'
pec(object,...)
## S3 method for class 'prodlm'
pec(object,...)
## S3 method for class 'survfit'
pec(object,...)
## S3 method for class 'aalen'
pec(object,...)

```

### Arguments

object	A named list of prediction models, where allowed entries are (1) R-objects for which a <a href="#">predictSurvProb</a> 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 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 <code>Surv(time, status)~x1+x2</code> and <code>cens.model="cox"</code> . Then the weights are based on a Cox regression model for the censoring times with predictors <code>x1</code> and <code>x2</code> . Note that the usual coding is assumed: <code>status=0</code> 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. <code>formula=Surv(time,status)~1</code> the <code>cens.model</code> is coerced to "marginal" and the Kaplan-Meier estimator for the censoring times is used to calculate the weights. If formula is missing, the formula of the first model in list is used.
data	A data frame in which to validate the prediction models and to fit the censoring model. If data is missing, the data of the first model in list is used.
times	A vector of timepoints. At each timepoint the prediction error curves are estimated. If <code>exact==TRUE</code> the times are merged with all the unique values of the response variable. If times is missing and <code>exact==TRUE</code> all the unique values of the response variable are used. If missing and <code>exact==FALSE</code> use an equidistant grid of values between <code>start</code> and <code>maxtime</code> . 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 <code>Surv</code> or <code>Hist</code> object then <code>start</code> defaults to 0, otherwise to the smallest observed value of the response variable. <code>start</code> 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.

<code>exact</code>	Logical. If TRUE estimate the prediction error curves at all the unique values of the response variable. If times are given and <code>exact=TRUE</code> then the times are merged with the unique values of the response variable.
<code>exactness</code>	An integer that determines how many equidistant gridpoints are used between start and maxtime. The default is 100.
<code>fillChar</code>	Symbol used to fill-in places where the values of the prediction error curves are not available. The default is NA.
<code>cens.model</code>	Method for estimating inverse probability of censoring weights: <code>cox</code> : A semi-parametric Cox proportional hazard model is fitted to the censoring times <code>marginal</code> : The Kaplan-Meier estimator for the censoring times <code>nonpar</code> : 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 <code>prodlim</code> and <code>neighborhood</code> from the <code>prodlim</code> package. <code>aalen</code> : The nonparametric Aalen additive model fitted to the censoring times. Requires the <code>timereg</code> package.
<code>ipcw.refit</code>	If TRUE the inverse probability of censoring weights are estimated separately in each training set during cross-validation.
<code>splitMethod</code>	SplitMethod for estimating the prediction error curves. <code>none/noPlan</code> : Assess the models in the same data where they are fitted. <code>boot</code> : DEPRECATED. <code>cvK</code> : K-fold cross-validation, i.e. <code>cv10</code> for 10-fold cross-validation. After splitting the data in K subsets, the prediction models (ie those specified in <code>object</code> ) 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. <code>BootCv</code> : 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. <code>Boot632</code> : Linear combination of <code>AppErr</code> and <code>BootCvErr</code> using the constant weight .632. <code>Boot632plus</code> : Linear combination of <code>AppErr</code> and <code>BootCv</code> using weights dependent on how the models perform in permuted data. <code>loocv</code> : Leave one out cross-validation. <code>NoInf</code> : Assess the models in permuted data.
<code>B</code>	Number of bootstrap samples. The default depends on argument <code>splitMethod</code> . When <code>splitMethod</code> in <code>c("BootCv","Boot632","Boot632plus")</code> the default is 100. For <code>splitMethod="cvK"</code> B is the number of cross-validation cycles, and – default is 1. For <code>splitMethod="none"</code> B is the number of bootstrap simulations e.g. to obtain bootstrap confidence limits – default is 0.

<code>M</code>	The size of the bootstrap samples for resampling without replacement. Ignored for resampling with replacement.
<code>reference</code>	Logical. If TRUE add the marginal Kaplan-Meier prediction model as a reference to the list of models.
<code>model.args</code>	List of extra arguments that can be passed to the <code>predictSurvProb</code> methods. The list must have an entry for each entry in object.
<code>model.parms</code>	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.
<code>keep.index</code>	Logical. If FALSE remove the bootstrap or cross-validation index from the output list which otherwise is included in the <code>splitMethod</code> part of the output list.
<code>keep.matrix</code>	Logical. If TRUE add all B prediction error curves from bootstrapping or cross-validation to the output.
<code>keep.models</code>	Logical. If TRUE keep the models in object. If "Call" keep only the call of these models.
<code>keep.residuals</code>	Logical. If TRUE keep the patient individual residuals at <code>testTimes</code> .
<code>keep.pvalues</code>	For <code>multiSplitTest</code> . If TRUE keep the pvalues from the single splits.
<code>noinf.permute</code>	If TRUE the noinformation error is approximated using permutation.
<code>multiSplitTest</code>	If TRUE the test proposed by van de Wiel et al. (2009) is applied. Requires subsampling bootstrap cross-validation, i.e. that <code>splitMethod</code> equals <code>bootcv</code> and that <code>M</code> is specified.
<code>testIBS</code>	A range of time points for testing differences between models in the integrated Brier scores.
<code>testTimes</code>	A vector of time points for testing differences between models in the time-point specific Brier scores.
<code>confInt</code>	Experimental.
<code>confLevel</code>	Experimental.
<code>verbose</code>	if TRUE report details of the progress, e.g. count the steps in cross-validation.
<code>savePath</code>	Place in your filesystem (directory) where training models fitted during cross-validation are saved. If missing training models are not saved.
<code>...</code>	Not used.

## Details

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`, `train.data`, ...

Such a function takes the object which was fitted with `train.data` 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)
prodlm from library(prodlm)
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:

<code>PredErr</code>	The estimated prediction error according to the <code>splitMethod</code> . A matrix where each column represents the estimated prediction error of a fit at the time points in time.
<code>AppErr</code>	The training error or apparent error obtained when the model(s) are evaluated in the same data where they were trained. Only if <code>splitMethod</code> is one of "NoInf", "cvK", "BootCv", "Boot632" or "Boot632plus".
<code>BootCvErr</code>	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 <code>splitMethod</code> is one of "Boot632" or "Boot632plus". When <code>splitMethod="BootCv"</code> then the <code>BootCvErr</code> is stored in the component <code>PredErr</code> .
<code>NoInfErr</code>	The prediction error when the model(s) are evaluated in the permuted data. Only if <code>splitMethod</code> is one of "BootCv", "Boot632", or "Boot632plus". For <code>splitMethod="NoInf"</code> the <code>NoInfErr</code> is stored in the component <code>PredErr</code> .
<code>weight</code>	The weight used to linear combine the <code>AppErr</code> and the <code>BootCvErr</code> Only if <code>splitMethod</code> is one of "Boot632", or "Boot632plus".
<code>overfit</code>	Estimated overfit of the model(s). See Efron & Tibshirani (1997, Journal of the American Statistical Association) and Gerds & Schumacher (2007, Biometrics). Only if <code>splitMethod</code> is one of "Boot632", or "Boot632plus".
<code>call</code>	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

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* (OnlineEarly Articles). 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.

### See Also

[plot.pec](#), [summary.pec](#), [R2](#), [crps](#)

### Examples

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

set.seed(130971)
dat <- SimSurv(300)

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

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

# 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=100,
                        verbose=TRUE)

print(PredError.632plus,times=seq(5,30,5))
summary(PredError.632plus)
plot(PredError.632plus,xlim=c(0,30))
## Not run:
# do the same again but now parallized
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=100,
                        verbose=TRUE)

## End(Not run)

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

## Not run:
library(survival)
library(compRisk)
library(cmprsk)
library(pec)
data(pbc)

```

```
f1 <- CRR(Hist(time,status)~sex+edema,cause=2,data=pbcc)
f2 <- CRR(Hist(time,status)~sex,cause=2,data=pbcc)
f3 <- compRisk(Hist(time,status)~sex+edema,cause=2,data=pbcc)
f4 <- compRisk(Hist(time,status)~sex+edema,cause=2,data=pbcc,link="prop")
p1 <- pec(list(f1,f2,f3,f4),formula=Hist(time,status)~1,data=pbcc,cause=2)

## End(Not run)
```

---

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	See cforest of the party package
data	See cforest of the party package
...	See cforest of the party package

### Details

See cforest of the party package.

### References

Mogensen, UB and Ishwaran, H and Gerds, TA (2010) Evaluating random forests for survival analysis using prediction error curves Research report 10/8. Department of Biostatistics, University of Copenhagen

---

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 <a href="#">pec</a> .
what	The name of the entry in x. Defaults 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 values are smoothed with the function <a href="#">smooth</a> kind="3R".
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 an artificial data frame
# with survival response and two predictors

set.seed(280180)
N <- 300
X1 <- rnorm(N,mean=10,sd=5)
X2 <- rbinom(N,1,.4)
linPred <- .00001+0.2*X1+2.3*X2
T <- sapply(linPred,function(lp){rexp(n=1,exp(-lp))})
C <- rexp(n=300,exp(-mean(linPred)))
dat <- data.frame(time=pmin(T,C),status=as.numeric(T<=C),X1=X1,X2=X2)

# 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),
"Cox.X2"=coxph(Surv(time,status)~X2,data=dat),
"Cox.X1.X2"=coxph(Surv(time,status)~X1+X2,data=dat))

# 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=10,
                        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,

```

---

plotPredictEventProb    *Plotting predicted survival curves.*

---

## Description

Plotting 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	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 <a href="#">SmartControl</a> and then passed to the functions: <a href="#">plot</a> , <a href="#">axis</a> , <a href="#">legend</a> .

### 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>

**See Also**

[predictEventProbprodlim](#)

**Examples**

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

---

```
plotPredictSurvProb    Plotting predicted survival curves.
```

---

**Description**

Plotting 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	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 <a href="#">SmartControl</a> and then passed to the functions: <a href="#">plot</a> , <a href="#">axis</a> , <a href="#">legend</a> .

### 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>

### See Also

[predictSurvProbprodlim](#)

### Examples

```
# generate some survival data
d <- SimSurv(100)
# then fit a Cox model
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(randomSurvivalForest)
rsfmodel <- rsf(Survrsf(time,status)~X1+X2,data=d)
plotPredictSurvProb(rsfmodel,newdata=ndat,times=ttt)
```

---

predictEventProb	<i>Predicting event probabilities (cumulative incidences) in competing risk models.</i>
------------------	---

---

### 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`.

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

### Usage

```
predictEventProb(object, newdata, times, cause, ...)
## S3 method for class 'CauseSpecificCox'
predictEventProb(object, newdata, times, cause, ...)
## S3 method for class 'riskRegression'
predictEventProb(object, newdata, times, cause, ...)
## S3 method for class 'FGR'
predictEventProb(object, newdata, times, cause, ...)
## S3 method for class 'prodlm'
predictEventProb(object, newdata, times, cause, ...)
```

### Arguments

<code>object</code>	A fitted model from which to extract predicted event probabilities
<code>newdata</code>	A data frame containing predictor variable combinations for which to compute predicted event probabilities.
<code>times</code>	A vector of times in the range of the response variable, for which the cumulative incidences event probabilities are computed.
<code>cause</code>	Identifies the cause of interest among the competing events.
<code>...</code>	Additional arguments that are passed on to the current method.

### Details

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](#).

---

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’.

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

**Usage**

```

predictSurvProb(object, newdata, times, ...)
## 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 'coxph'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'cph'
predictSurvProb(object, newdata, times,...)
## Default S3 method:
predictSurvProb(object, newdata, times,...)
## S3 method for class 'rsf'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'matrix'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'mfp'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'pecCforest'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'prodlm'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'psm'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'selectCox'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'survfit'
predictSurvProb(object, newdata, times,...)
## S3 method for class 'phnnet'

```

```

predictSurvProb(object, newdata, times, train.data, ...)
## S3 method for class 'survnnnet'
predictSurvProb(object, newdata, times, train.data, ...)
## S3 method for class 'rpart'
predictSurvProb(object, newdata, times, train.data, ...)

```

### 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.
train.data	An optional data frame which contains the response and predictor variable combinations in which the prediction model was trained
...	Additional arguments that are passed on to the current method.

### Details

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`), `mfp` (library `mfp`), `phnnet` (library `survnnnet`), `survnnnet` (library `survnnnet`), `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 suppressing 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>

### See Also

[predict,survfit](#)

**Examples**

```
# generate some survival data
d <- SimSurv(100)
# then fit a Cox model
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
predictSurvProb(coxmodel,newdata=ndat,times=ttt)

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

---

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, ...)
## S3 method for class 'pec'
summary(object,times,what,models,digits=3,print=TRUE,...)
```

**Arguments**

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

**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 <a href="#">pec</a>
models	For which of the models in <code>object\$models</code> should we compute $R^2(t)$ . By default all models are used except for the reference model.
what	The name of the entry in <code>x</code> to be used. Defaults to <code>PredErr</code> Other choices are <code>AppErr</code> , <code>BootCvErr</code> , <code>Boot632</code> , <code>Boot632plus</code> .
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)

dat=prodlm::SimSurv(100)
nullmodel=prodlm(Hist(time,status)~1,data=dat)
pmodel1=coxph(Surv(time,status)~X1+X2,data=dat)
pmodel2=coxph(Surv(time,status)~X2,data=dat)
perror=pec(list(Cox1=pmodel1,Cox2=pmodel2),Hist(time,status)~1,data=dat,reference=TRUE)
R2(perror,times=seq(0,1,.1),reference=1)
```

---

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 <a href="#">fastbw</a> for details.

**Details**

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

**References**

Mogensen, UB and Ishwaran, H and Gerds, TA (2010) Evaluating random forests for survival analysis using prediction error curves Research report 10/8. Department of Biostatistics, University of Copenhagen

**Examples**

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

---

Special	<i>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.</i>
---------	--

---

**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**

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

# Index

- \*Topic **datasets**
  - GBSG2, 8
- \*Topic **prediction**
  - resolvesplitMethod, 30
- \*Topic **survival**
  - cindex, 2
  - crps, 6
  - ipcw, 9
  - pec, 11
  - pecCforest, 18
  - plot.pec, 18
  - plotPredictEventProb, 21
  - plotPredictSurvProb, 23
  - predictEventProb, 25
  - predictSurvProb, 26
  - print.pec, 28
  - R2, 29
  - selectCox, 31
- axis, 22, 24
- cindex, 2
- crps, 6, 16
- fastbw, 31
- GBSG2, 8
- ibs (crps), 6
- ipcw, 9
- legend, 22, 24
- pec, 7, 10, 11, 19, 20, 29, 30
- pecCforest, 18
- plot, 20, 22, 24
- plot.pec, 16, 18, 33
- plotPredictEventProb, 21
- plotPredictSurvProb, 23
- predict, 27
- predictEventProb, 23, 25
- predictSurvProb, 3, 12, 24, 25, 26, 26
- print.pec, 28
- prodlim, 20, 23, 24
- R2, 16, 29
- resolvesplitMethod, 30
- selectCox, 31
- SmartControl, 22, 24
- smooth, 19
- Special, 20, 32
- summary.pec, 16, 20
- summary.pec (print.pec), 28
- survfit, 27