

Package ‘pec’

June 24, 2009

Title Prediction Error Curves for Survival Models

Version 1.1.1

Author Thomas A. Gerds

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

Depends R (>= 1.9.1), prodlim, survival

Suggests Design, timereg, mfp, survnnet, rpart

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

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-06-24 14:04:41

R topics documented:

GBSG2	2
ipcw	3
pec	4
plot.pec	10
predictSurvProb	12
print.pec	14
R2	15
Index	17

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 2×2 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,  
      model = c("cox", "marginal", "nonpar", "aalen", "none"),  
      times,  
      otimes)
```

Arguments

<code>formula</code>	A survival formula like, <code>Surv(time,status)~1</code> , where as usual <code>status=0</code> means censored. The status variable is internally reversed for estimation of censoring rather than survival probabilities. Some of the available models (see argument <code>model</code>) will use predictors on the right hand side of the formula.
<code>data</code>	The data used for fitting the censoring model
<code>model</code>	Censoring model used for estimation of the (conditional) censoring distribution.
<code>times</code>	Time values at which the probabilities of not being censored are returned
<code>otimes</code>	The time values, usually the original event times in the <code>data.frame</code> <code>data</code> , at which lagged individual probabilities of not being censored are returned.

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.

Currently the data set should be order by time to get the value `wt.obs` in the right order for some choices of `model`!

Value

<code>wt</code>	Estimated weights at <code>times</code>
<code>wt.obs</code>	Estimated weights at individual time values <code>otimes</code>
<code>fit</code>	The fitted censoring model

Author(s)

Thomas A. Gerds (tag@biostat.ku.dk)

See Also[pec](#)**Examples**

```
## Not run:
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,model="marginal",times=sort(unique(dat$time)),otimes=
plot(WKM$fit)
WKM$fit

# using the Cox model for the censoring times given edema

WCox=ipcw(Surv(time,status)~X2,data=dat,model="cox",times=sort(unique(dat$time)),otimes=dat$
WCox$fit

plot(WKM$fit)
lines(sort(unique(dat$time)),1-WCox$wt[1,],type="l",col=2,lty=3,lwd=3)
lines(sort(unique(dat$time)),1-WCox$wt[2,],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,model="nonpar",times=sort(unique(dat$time)),otimes=d
plot(WKM2$fit,add=TRUE)
## End(Not run)
```

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 and they might depend on covariates. Prediction error curves are obtained when the Brier score is followed over time. Bootstrap-crossvalidation techniques are implemented to estimate the generalization error.

Usage

```
pec(object,...)
## S3 method for class 'list':
pec(object,
```

```

    formula,
    data,
    times,
    start,
    maxtime,
    exact=TRUE,
    exactness=100,
    fillChar=NA,
    cens.model="cox",
    replan="none",
    B,
    M,
    model.args=NULL,
    model.parms=NULL,
    keep.matrix=FALSE,
    import=NULL,
    export=NULL,
    na.accept=0,
    verbose=TRUE,
    ...)
## 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. Each entry is either an R-object for which a predictSurvProb method exists (see details) or a call that can be evaluated to such an R-object. For resampling (e.g. when <code>replan</code> is "boot632plus") all the models in this list must include their call in their value.
formula	A formula. If missing the formula of the first model in list is used. The left hand side is used to find the status response variable in <code>data</code> . 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)~x+y</code> and <code>cens.model="cox"</code> . Then the weights are based on a Cox regression model for the censoring times with predictors 'x' and 'y'. 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 <code>data</code> . 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.
<code>data</code>	A data frame in which to validate the prediction models and to fit the censoring model. If missing the data of the first model in list is used.
<code>times</code>	A vector of timepoints. At each timepoint the prediction error curves are estimated. If <code>exact==TRUE</code> the <code>times</code> are merged with all the unique values of the response variable. If <code>times</code> 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 <code>exactness</code> .
<code>start</code>	Minimal time for estimating the prediction error curves. If missing and <code>formula</code> 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 <code>times</code> are given.
<code>maxtime</code>	Maximal time for estimating the prediction error curves. If missing the largest value of the response variable is used.
<code>exact</code>	Logical. If <code>TRUE</code> estimate the prediction error curves at all the unique values of the response variable. If <code>times</code> are given and <code>exact=TRUE</code> then the <code>times</code> are merged with the unique values of the response variable.
<code>exactness</code>	An integer that determines how many equidistant gridpoints are used between <code>start</code> and <code>maxtime</code> . 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 <code>NA</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>replan</code>	Method for estimating the prediction error curves. <code>none</code> : Assess the models in the same data where they are fitted. <code>boot</code> : Bootstrap the prediction error curves <code>B</code> times WITHOUT cross-validation. <code>cvK</code> : <code>K</code> -fold cross-validation, i.e. <code>cv10</code> for 10-fold cross-validation. After splitting the data in <code>K</code> subsets, the prediction models (ie those specified in <code>object</code>) are evaluated on the data omitting the <code>K</code> th subset (training step). The prediction error is estimated with the <code>K</code> th subset (validation step). The random splitting is repeated <code>B</code> times and the estimated prediction error curves are obtained by averaging. <code>OutOfBag</code> : Bootstrap cross validation. The prediction models are trained on <code>B</code> bootstrap samples, that are either drawn with 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>OutOfBag</code> using the constant weight <code>.632</code> .
	<code>Boot632plus</code> : Linear combination of <code>AppErr</code> and <code>OutOfBag</code> using weights dependent on how the models perform in permuted data.
	<code>NoInf</code> : Assess the models in permuted data.
<code>B</code>	Number of bootstrap samples. The default depends on argument <code>replan</code> . When <code>replan</code> in <code>c("OutOfBag", "Boot632", "Boot632plus")</code> the default is 100. For <code>replan="cvK"</code> <code>B</code> is the number of cross-validation cycles, and <code>-</code> default is 1. For <code>replan="none"</code> <code>B</code> 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.
<code>model.args</code>	Experimental. List of extra arguments that can be passed to the <code>predictSurvProb</code> methods. The list must have an entry for each entry in <code>object</code> .
<code>model.parms</code>	Experimental. List of with exactly one entry for each entry in <code>object</code> . Each entry names parts of the value of the fitted models that should be extracted and added to the value.
<code>keep.matrix</code>	Logical. If <code>TRUE</code> add all <code>B</code> prediction error curves to the output.
<code>import</code>	Experimental. Import the bootstrap matrix from an external file. <code>import</code> is a list of arguments that passed to internal function <code>pecImportIndex</code>
<code>export</code>	Experimental. Export the bootstrap matrix to an external file. <code>export</code> is a list of arguments that passed to internal function <code>pecExportIndex</code> together with the matrix.
<code>na.accept</code>	Only for "OutOfBag" error. The maximal number of failures during training the models to the bootstrap samples. Usually a small number relative to <code>B</code> .
<code>verbose</code>	if <code>TRUE</code> the procedure is reporting details of the progress, e.g. it prints the current step in resampling procedures.
<code>...</code>	Difficult to explain

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(Design)
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:

<code>PredErr</code>	The estimated prediction error according to the <code>replan</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>replan</code> is one of "NoInf", "cvK", "OutOfBag", "Boot632" or "Boot632plus".
<code>OutOfBagErr</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>replan</code> is one of "Boot632" or "Boot632plus". When <code>replan="OutOfBag"</code> then the <code>OutOfBagErr</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>replan</code> is one of "OutOfBag", "Boot632", or "Boot632plus". For <code>replan="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>OutOfBagErr</code> Only if <code>replan</code> is one of "Boot632", or "Boot632plus".
<code>overfit</code>	Estimated <code>overfit</code> of the model(s). See Efron & Tibshirani (1997, Journal of the American Statistical Association) and Gerds & Schumacher (2007, Biometrics). Only if <code>replan</code> is one of "Boot632", or "Boot632plus".
<code>call</code>	The call that produced the object
<code>time</code>	The time points at which the prediction error curves change.

<code>ipcw.fit</code>	The fitted censoring model that was used for re-weighting the Brier score residuals. See Gerds & Schumacher (2006, Biometrical Journal)
<code>n.risk</code>	The number of subjects at risk for all time points.
<code>models</code>	The prediction models fitted in their own data.
<code>cens.model</code>	The censoring models.
<code>maxtime</code>	The latest timepoint where the prediction error curves are estimated.
<code>start</code>	The earliest timepoint where the prediction error curves are estimated.
<code>exact</code>	TRUE if the prediction error curves are estimated at all unique values of the response in the full data.
<code>method</code>	The method used for estimation of the prediction error.

Author(s)

Thomas 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)
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 apparent prediction error
PredError <- pec(object=Models,
                 formula=Surv(time,status)~X1+X2,
                 data=dat,
                 exact=TRUE,
                 cens.model="marginal",
                 replan="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",
                        replan="boot632plus",
                        B=100,
                        verbose=TRUE)

print(PredError.632plus,times=seq(5,30,5))
summary(PredError.632plus)
plot(PredError.632plus,xlim=c(0,30))

```

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="PredErr",

```

```

who,
xlim=c(x$start, x$maxtime),
ylim=c(0, 0.3),
xlab="Time",
ylab,
lwd.lines=2,
axes=TRUE,
col,
lty,
lines.type,
smooth=FALSE,
add.refline=FALSE,
add=FALSE,
legend=ifelse(add, FALSE, TRUE),
legend.text,
legend.args=list(),
specials=NULL,
...)

```

Arguments

x	object of class <code>pec</code> obtained with function <code>pec</code>
what	the name of the entry in <code>x</code> Defaults to <code>PredErr</code> Other choices are <code>AppErr</code> , <code>OutOfBagErr</code> , <code>NoInfErr</code> , <code>overfit</code>
who	Specifies models in <code>x\$models</code> 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.
lwd.lines	Extra parameter passed to function <code>lines</code> .
axes	if <code>FALSE</code> no axes are drawn
col	vector of colors given to the curves of models in the order determined by <code>who</code>
lty	vector of <code>lty</code> 's given to the curves of models in the order determined by <code>who</code>
lines.type	passed to <code>lines</code>
smooth	if <code>TRUE</code> the plotting values are smoothed with the function <code>smooth</code> <code>kind="3R"</code>
add.refline	if <code>TRUE</code> 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	<code>TRUE</code> means that only lines are added to an existing device
legend	<code>TRUE</code> means draw the legend
legend.text	legend text in the order determined by <code>who</code>
legend.args	a list with arguments that are passed to <code>legend</code> .
specials	Experimental
...	extra arguments that are passed to <code>plot</code> .

Author(s)

Thomas Gerds (tag@biostat.ku.dk)

See Also

[pec](#)

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 '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 'glm':
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 '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 'phnnet':
predictSurvProb(object, newdata, times, train.data, ...)
## S3 method for class 'survnnet':

```

```

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

```

Arguments

<code>object</code>	A model for which predicted probabilities are desired.
<code>newdata</code>	A data frame containing predictor variable combinations for which predictions are desired
<code>times</code>	A vector of times in the range of the response variable, e.g. times when the response is a survival object, at which the exceedance probabilities (i.e. the survival probabilities) are returned.
<code>train.data</code>	An optional data frame which contains the response and predictor variable combinations in which the prediction model was trained
<code>...</code>	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 `Design`), `coxph` (library `survival`), `aalen` (library `timereg`), `cox.aalen` (library `timereg`), `mfp` (library `mfp`), `phnnet` (library `survnnet`), `survnnet` (library `survnnet`), `rpart` (library `rpart`), `product.limit` (library `prodlim`), `survfit` (library `survival`), `psm` (library `Design`), `glm` (library `stats`).

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](#)

 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. By default the estimate is shown which corresponds to the <code>replan</code> argument in the call to <code>pec</code> . This is stored in <code>predErr</code> . For <code>replan</code> in <code>c("cvk", "bootB0", "boot.632", "boot.632plus")</code> also the apparent error <code>AppErr</code> is a possible choice. If <code>replan</code> in <code>c("boot.632plus", "boot.632plus", "boot.632plus", "boot.632plus")</code> then also the out of bag error (<code>OutOfBagErr</code>), the estimated overfit (<code>overfit</code>), and the no-information-error (<code>NoInfErr</code>) are possible choices.
models	Which models in the list <code>object\$models</code> should be shown. Defaults to all models.
digits	Number of decimals used in tables.
print	Set to <code>FALSE</code> 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](#)

Description

Cumulating prediction error curves over time and a time-dependent R^2 like measure.

Usage

```
R2(object, who, what, times, nullModel=1)
crps(object, who, what, times, start)
```

Arguments

<code>object</code>	An object with estimated prediction error curves obtained with the function pec
<code>who</code>	Which models in <code>object\$models</code> should be considered. Default is all models for <code>crps</code> and all models except the reference for <code>R2</code>
<code>what</code>	Usually <code>pred.error</code> – if crossvalidation or bootstrap methods are used then also the <code>apparent.error</code> and if <code>replan=boot.632</code> plus then also the <code>BootB0.error</code> , the estimated overfit, and the <code>NoInf.error</code> can be cumulated or compared to a reference prediction model.
<code>times</code>	Time points at which the summaries are shown.
<code>start</code>	Only for <code>crps</code> : the time point at which the cumulation is started
<code>nullModel</code>	Position of the model whose prediction error is used as the reference in the denominator when constructing R^2

Details

The cumulative prediction error (continuous ranked probability score) is defined as the area under the prediction error curve.

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 with a column for every requested prediction model

Author(s)

Thomas A. Gerds (tag@biostat.ku.dk)

See Also

[pec](#)

Examples

```
set.seed(18713)

## Not run:
dat=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
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)

R2(perror,times=seq(0,1,.1))
## End(Not run)
```

Index

*Topic **datasets**

GBSG2, 1

*Topic **survival**

ipcw, 2

pec, 4

plot.pec, 10

predictSurvProb, 12

print.pec, 14

R2, 15

crps, 9

crps (R2), 15

GBSG2, 1

ipcw, 2

legend, 11

lines, 11

pec, 3, 4, 11, 14, 15

plot, 11

plot.pec, 9, 10

predict, 13

predictSurvProb, 5, 12

predictSurvProb.cox.aalen
(*predictSurvProb*), 12

predictSurvProb.coxph
(*predictSurvProb*), 12

predictSurvProb.cph
(*predictSurvProb*), 12

predictSurvProb.default
(*predictSurvProb*), 12

predictSurvProb.glm
(*predictSurvProb*), 12

predictSurvProb.matrix
(*predictSurvProb*), 12

predictSurvProb.mfp
(*predictSurvProb*), 12

predictSurvProb.phnnet
(*predictSurvProb*), 12

predictSurvProb.prodlm
(*predictSurvProb*), 12

predictSurvProb.psm
(*predictSurvProb*), 12

predictSurvProb.rpart
(*predictSurvProb*), 12

predictSurvProb.survfit
(*predictSurvProb*), 12

predictSurvProb.survnnnet
(*predictSurvProb*), 12

print.pec, 14

R2, 9, 15

smooth, 11

summary.pec, 9

summary.pec (*print.pec*), 14

survfit, 13