Evaluating random forests for survival analysis using prediction error curves

Ulla B. Mogensen
Hemant Ishwaran
Thomas A. Gerds

Research Report 10/8
Department of Biostatistics
University of Copenhagen
Evaluating random forests for survival analysis using prediction error curves

Ulla B Mogensen, Hemant Ishwaran, Thomas A Gerds

Abstract

Prediction error curves are increasingly used to assess and compare predictions in survival analysis. This article surveys the R package pec which provides a set of functions for efficient and simultaneous computation of prediction error curves for a list of rival prediction models selected by a regression modeling strategy (rms) or a machine learning method. The routines implement inverse probability of censoring weights to deal with right censored data and bootstrap cross-validation to deal with the apparent error problem.

Random forests is a nonparametric machine learning strategy that can be used for building a prediction model in survival analysis. This article describes how to compute prediction error curves for random forest prediction models in survival analysis derived from two different R-packages (randomSurvivalForest, cforest). The functionality of the pec package is illustrated using the data of the Copenhagen Stroke Study.

Keywords: Survival prediction, prediction error curves, random survival forest, R.

1. Introduction

In survival analysis many different regression modeling strategies can be applied to predict the risk of future events. Often, however, the default choice of analysis tends to rely on Cox regression modeling due to its convenience. Extensions of the random forest approach (Breiman 2001) to survival analysis provide an alternative way to build a risk prediction model. This bypasses the need to impose parametric constraints on the underlying distributions and provides a way to automatically deal with high-level interactions and higher-order terms in variables and allows for accurate prediction (Ishwaran, Kogalur, Blackstone, and Lauer 2008).

In this article we present the R-package pec, short for Prediction Error Curves. Prediction error curves are time dependent estimates of the population average Brier score. For a vector of time points, the Brier score for a single subject is defined as the squared difference of observed survival status (e.g., 1=alive and 0=dead) to that of model based predictions of survival
probabilities. To avoid bias for right censored event times the pec function implements inverse probability of censoring weights (Graf, Schmoor, Sauerbrei, and Schumacher 1999; Gerds and Schumacher 2006). Data splitting into independent training and test set, i.e. cross-validation, is implemented in a general way for the typical situation where a single data set has to be used to build the prediction models and again to estimate the prediction error (Efron and Tibshirani 1997; Gerds and Schumacher 2007). It is also possible to use the pec function to compute prediction error curves if an independent test data set is available and no additional data splitting is wanted.

An important feature of the pec function is that the entire model building process can be evaluated, including data dependent steps such as variable selection, shrinkage, or estimation of link functions. Consequently, when repeated data splitting is used, the resulting estimates of the prediction error are a composite of the prediction accuracy and the underlying variability of the prediction models due to whatever data dependent steps were used for their construction over the training splits of the data.

To illustrate the usage of pec we have extended the package to work with prediction models obtained with the functions of the R-packages randomSurvivalForest (Ishwaran and Kogalur 2007; Ishwaran et al. 2008) and party (Hothorn, Bühlmann, Dudoit, Molinaro, and van der Laan 2006) which implement random forest analyses for survival data. The new functions are illustrated in a worked out example where we analyse the data of the Copenhagen Stroke Study (COST) (Jørgensen, Nakayama, Raaschou, Gam, and Olsen 1994). Earlier analyses of COST were based on a Cox regression model where the predictor variables were selected by backward selection (Andersen, Andersen, Kammersgaard, and Olsen 2005). Here we compare strategies based on random forests to an automated model finding strategy which first selects predictor variables based on stepwise backward selection and then predicts the survival chances based on a Cox regression model which includes the final selection of predictor variables.

### 2. Predicting survival

#### 2.1. Data structure

A survival prediction model requires data on the life history of subjects (the response) and their characteristics (the predictor variables). The response is a matrix with two columns: the minimum $\tilde{T}_i$ of the survival time $T_i$ and the right censoring time $C_i$ and the status variable $\Delta_i = I\{T_i \leq C_i\}$. The vector of predictor variables $X_i = (X_{i1}, \ldots, X_{ik})$ for subject $i$ usually consists of a mixture of continuous scale variables, like age or blood pressure, and dummy coding of qualitative variables, like gender or genotype.

**Example** We reconsider the data of the Copenhagen Stroke Study (COST) (Jørgensen et al. 1994). In the COST study 993 patients were enrolled after being admitted to a hospital with a stroke and were followed for 10 years. Recorded for each patient was the length of time from admission to death, for those who died, otherwise the length of time from admission to the maximal time where the patient was known to be alive was recorded (i.e., right censored). The following output shows the first six lines of the observed survival response:

```r
R> head(cost[,c("time","status")])
```
For example, patients 2 and 3 died 1369 and 1657 days after admission, respectively, while patient 5 was right censored at 3670 days (thus we know the patient survived at least 3670 days). Table 1 shows the distribution of the 13 predictor variables that were collected as part of the study.

[Table 1 about here.]

For the purpose of illustration we construct three hypothetical new patients with values in the range of the predictor space defined by the COST patients and store them in a data frame called `newData`. These new “patients” have different ages, but do not differ otherwise: all are females, have a strokeScore of 38, a mean value of 6 for cholest, and the remaining predictor variables set to the value “no”.

2.2. Stepwise variable selection in Cox regression

A Cox regression model specifies the conditional cumulative hazard function dependent on the vector of predictor variables $X_i = (X_i^1, \ldots, X_i^K)$:

$$\Lambda(t|X_i) = \Lambda_0(t) \exp \left( \beta^T X_i \right).$$

Here $\Lambda_0$ is an unspecified increasing function, referred to as the cumulative baseline hazard and $\beta = (\beta_1, \ldots, \beta_K)$ is a vector of regression coefficients.

Many different variable selection strategies can be applied within the context of Cox regression. Here we consider stepwise variable selection as implemented in the function `fastbw` of the `rms` package (Harrell Jr 2009). More specifically we use the option of the Akaike information criteria (AIC) in the first step of the following algorithm.

**Step 1.** Run `fastbw` and save the list of predictor variables which have corresponding nonzero regression coefficients.

**Step 2.** Fit a Cox regression model that includes the predictor variables selected in **Step 1**, e.g. with the function `cph` from the `rms` package.

**Step 3.** Using the estimates of $\beta$ and $\Lambda_0$ obtained in **Step 2**, calculate for a subject with predictor values $x$:

$$\hat{S}^{\text{cox}}(t|x) = \exp \left( -\hat{\Lambda}_0(t) \exp \left[ \hat{\beta}^T x \right] \right).$$

2.3. Random forests
A random forest is a non-parametric machine learning strategy that can be used for building a risk prediction model in survival analysis. In survival settings, the predictor is an ensemble formed by combining the results of many survival trees. The general strategy is as follows:

**Step 1.** Draw $B$ bootstrap samples.

**Step 2.** Grow a survival tree based on the data of each of the bootstrap samples $b = 1, \ldots, B$:

(a) Select a subset of the predictor variables.

(b) Among all binary splits defined by the predictor variables selected in (a), find the best split of the current bootstrap sample into two subsets (the daughter nodes) according to a suitable criterion for right censored data, like the log-rank test.

(c) Repeat (a)-(b) with the daughter nodes until a stopping criterion is met.

**Step 3.** Aggregate information from the terminal nodes (nodes with no further split) from the $B$ survival trees to obtain a risk prediction ensemble.

In what follows we consider two different implementations of random forests from the two R packages: `randomSurvivalForest` (Ishwaran et al. 2008) and `party` (Hothorn et al. 2006).

To describe the risk predictions for the forest ensembles, denote for the $b$th survival tree $T_b(x)$ as the terminal node of subjects in the $b$th bootstrap sample in which a subject with predictor values $x$ ends up. Note that when bootstrap samples are drawn with replacement then subjects from the original data set may occur multiple times. Thus, let $c_{ib}$ be the number of times subject $i$ occurs in the $b$th bootstrap sample. If the $i$th subject is not in the bootstrap sample then $c_{ib} = 0$. Furthermore, introduce the following counting process notation (Andersen, Borgan, Gill, and Keiding 1993)

$$N_i(s) = \mathbb{I}(\hat{T}_i \leq s, \Delta_i = 1); \quad Y_i(s) = \mathbb{I}(\hat{T}_i > s).$$

and

$$N^*_b(s, x) = \sum_{i=1}^N c_{ib} \mathbb{I}(X_i \in T_b(x)) N_i(s); \quad Y^*_b(s, x) = \sum_{i=1}^N c_{ib} \mathbb{I}(X_i \in T_b(x)) Y_i(s).$$

**randomSurvivalForest**

In random survival forests (Ishwaran et al. 2008), the ensemble is constructed by aggregating tree-based Nelson-Aalen estimators. Specifically, in each terminal node of a tree, the conditional cumulative hazard function is estimated using the Nelson-Aalen estimator for those subjects that are in the bootstrap sample (i.e, the “in-bag” data):

$$\hat{H}_b(t|x) = \int_0^t \frac{N^*_b(ds, x)}{Y^*_b(s, x)}.$$

The survival prediction from the random survival forest at $x$ is then obtained as

$$\hat{S}^{rsf}(t|x) = \exp \left( -\frac{1}{B} \sum_{b=1}^B \hat{H}_b(t|x) \right).$$

(1)
Party

The conditional inference forest for survival analysis (Hothorn, Lausen, Benner, and Radespiel-Tröger 2004) is to predict with a weighted Kaplan-Meier estimate based on all subjects from the training data at \( x \) as follows:

\[
\hat{S}_{\text{cforest}}(t|x) = \prod_{s \leq t} \left(1 - \frac{\sum_{b=1}^{B} N_b^*(ds, x)}{\sum_{b=1}^{B} Y_b^*(s, x)}\right)
\]  

(2)

If the underlying survival function is continuous, then this is asymptotically equivalent to

\[
\exp \left(- \int_0^t \frac{\sum_{b=1}^{B} N_b^*(ds, x)}{\sum_{b=1}^{B} Y_b^*(s, x)} \right)
\]

which by comparison to equation (1) shows that way trees are aggregated is different from the random survival forest approach.

3. Writing pec extensions

The S3-methods predictSurvProb.x extract survival probabilities from R objects of class x. Previous versions of the pec package implemented methods for the following classes (package): matrix (base), aalen (timereg), cox.aalen (timereg), mfp (mfp), coxph (survival), survnnet (survnnet), rpart (rpart), cph (rms), survfit (survfit), prodlim (prodlim).

Below we explain how to extend the package to work with R objects of classes fastbw (rms), rsf (randomSurvivalForest), and cforest (party). At present, only S3-objects are allowed, but it is relatively easy to wrap S4-objects into the required form, as shown below for cforest. A requirement for repeated data splitting (cross-validation) is that the R object contains its call, that has to be a list in which the argument data can be modified.

A predictSurvProb method has three required arguments:

- **object**: the fitted R object
- **newdata**: a data frame with the predictor variables
- **times**: a vector of time points.

The functions then extract survival probabilities from the object and returns them in a matrix with as many columns as there are times and as many rows as there are lines in newdata. It is possible to pass further arguments to the predictSurvProb method via the argument model.args of the function pec. For example the function predictSurvProb.rpart uses an optional argument train.data.

3.1. Selected Cox regression

The following function selectCox evaluates Step 1 and Step 2 the stepwise variable selection strategy for the Cox regression model described in Section 2.2.
Evaluating random forests for survival analysis using prediction error curves

```r
selectCox <- function(formula, data, rule = "aic"){
  require(rms)
  require(prodlim)
  fit <- cph(formula, data, surv=TRUE)
  bwfit <- fastbw(fit, rule=rule)
  if (length(bwfit$names.kept)==0){
    newform <- reformulate("1", formula[[2]])
    newfit <- prodlim(newform, data=data)
  } else{
    newform <- reformulate(bwfit$names.kept, formula[[2]])
    newfit <- cph(newform, data, surv=TRUE)
  }
  out <- list(fit=newfit, In=bwfit$names.kept)
  out$call <- match.call()
  class(out) <- "selectCox"
  out
}
```

The function `cph` from `rms` is used to fit a Cox regression model using the selected predictor variables, with one exception: If the set of selected predictor variables in Step 1 is empty, then the Kaplan-Meier method is applied to predict survival via the function `prodlim` (`prodlim`). The resulting R object is assigned to the S3-class `selectCox`. Evaluation of Step 3 in Section 2.2 is done with the following `predictSurvProb` method which passes the arguments either to the `predictSurvProb.cph` or the `predictSurvProb.prodlim` method, depending on whether or not the selected set of predictor variables was empty.

```r
predictSurvProb.selectCox <- function(object, newdata, times, ...)
{
  predictSurvProb(object[[1]], newdata=newdata, times=times, ...)
}
```

3.2. Random survival forest package: rsf

A random survival forest model is fitted with the function `rsf` (`randomSurvivalForest`) which results in an object of S3-class `rsf`. Using the built-in `predict.rsf` method we extract the averaged cumulative hazard function for each line in `newdata` at the event times of the original data set (see Section 2.3). The survival probabilities are then computed via formula (1) and with the help of the function `sindex` (`prodlim`) these are evaluated at the requested times.

```r
predictSurvProb.rsf <- function(object, newdata, times, ...)
{
  require(randomSurvivalForest)
  H <- predict.rsf(object=object, test=newdata)$ensemble
  S <- exp(-H)
  Time <- object$timeInterest
  p <- cbind(1, S)[, 1 + sindex(jump.times=Time, eval.times=times), drop=FALSE]
  p
}
```
3.3. Party package: cforest

A conditional inference forest model is fitted with the function `cforest` (party) and results in an S4-class object. We get around the class problem by creating a wrapper function `pecCforest`. The fitted `cforest` S4-class object is stored in a list which is supplied with the call. The output is assigned to the S3-class `pecCforest`.

```r
cCforest <- function(formula, data, ...) {
  require(party)
  out <- list(forest = cforest(formula, data, ...))
  class(out) <- "pecCforest"
  out$call <- match.call()
  out
}
```

The `treeresponse` method from the `party` package can be applied to the list element `pecCforest$forest` of the S3-class object in order to extract survival probabilities for `newdata` at `times` (see formula (2)). The resulting object is a list which contains for each line in `newdata` the Kaplan-Meier curve in form of a `survfit` object (`survival`). We then apply `predictSurvProb.survfit` to the elements of the list.

```r
predictSurvProb.pecCforest <- function(object, newdata, times, ...) {
  require(party)
  survObj <- treeresponse(object$forest, newdata = newdata)
  p <- do.call("rbind", lapply(survObj, function(x) {
    predictSurvProb(x, newdata = newdata[1, , drop = FALSE], times = times)
  })))
  p
}
```

Example (continued) We use the complete case data of the COST study that contains data from 518 patients with no missing values in any of the 13 predictor variables.

```r
R> cost = na.omit(cost)
```

We fit a random survival forest model based on 1000 trees under default settings of the package (Ishwaran and Kogalur 2007). We also fit a conditional inference forest model via `pecCforest` based on 1000 trees, otherwise using the default options. Finally a selected Cox regression model is fitted via `selectCox` as described above.

```r
R> fitform = Surv(time, status) ~ age + sex + hypTen + myoInf + prevStroke + othDisease + alcohol + diabetes + smoke + hemor + intClau + strokeScore + cholest
R> fitcox = selectCox(fitform, data = cost, rule = "aic")
R> set.seed(13)
R> fitrsf = rsf(fitform, data = cost, forest = TRUE, ntree = 1000)
R> set.seed(13)
R> fitcforest <- pecCforest(fitform, data = cost, controls = cforest_classical(ntree = 1000))
```
Evaluating random forests for survival analysis using prediction error curves

To illustrate the results we predict the 10 year survival probabilities for the hypothetical new patients stored in the R object `newData` (see Section 2.1).

\begin{verbatim}
R> pcox=predictSurvProb(fitcox,newdata=newData,times=10*365.25)
R> prsf=predictSurvProb(fitrsf,newdata=newData,times=10*365.25)
R> extends <- function(...)TRUE
R> pcf=predictSurvProb(fitcforest,newdata=newData,times=10*365.25)
\end{verbatim}

The function `extends` that is set to TRUE ensures `cforest` that we predict a survival probability. Not necessary if one uses the functions only in `party` package.

Table 2 shows the results.

Compared to the selected Cox regression model both random survival forest approaches predict less extreme on the boundary of the age range (rows 1 and 3) at 10 year survival. This may be explained by the fact that a random forest is a “nearest neighbor type” method whereas a Cox regression model extrapolates the trend found in the center of the age range. Interestingly, for all `newData` the conditional inference forest model predicts a lower 10 year survival chance than the random survival forest model.

We use the function `plotPredictSurvProb` to plot the predicted survival curves for new subjects based on a fitted modeling strategy. It applies the `predictSurvProb` method to predict at all event times but other time points can be selected. The following code produces the curves in Figure 1.

\begin{verbatim}
R> par(mfrow=c(1,3))
R> lapply(1:3, function(x){
+   plotPredictSurvProb(fitcox,newdata=newData[x,],lty=1)
+   plotPredictSurvProb(fitrsf, newdata=newData[c(x,x),],add=TRUE,lty=2)
+   plotPredictSurvProb(fitcforest, newdata=newData[x,], add=TRUE, lty=3)
+ })
\end{verbatim}

Figure 1 shows the survival curves obtained with the rival modeling strategies for the hypothetical new patients in `newData`. The three models predict similar survival curves at median age but differently for the young and the old hypothetical patient where in both cases the selected Cox regression model is more extreme.

4. Prediction error curves

The function `pec` estimates and compares the predictive performances of rival survival modeling strategies over time. The prediction error at time point \( t \) is defined as the expected Brier score

\[ BS(t, \hat{S}) = E(Y_i(t) - \hat{S}(t|X_i))^2, \]
where \( i \) is a subject not in the training data, \( Y_i(t) = \mathcal{I}(T_i \geq t) \) is the true status of subject \( i \) and \( \hat{S}(t|X_i) \) is the predicted survival probability at time \( t \) for subject \( i \) with predictor variables \( X_i \). Useful benchmark values for the Brier score are 33\%, which corresponds to predicting the risk by a random number drawn from \( U[0, 1] \), and 25\% which corresponds to predicting 50\% risk for everyone. The most important benchmark is the expected Brier score of a prediction model which ignores all predictor variables. In survival analysis the Kaplan-Meier method calculated with all training samples yields such a null model.

4.1. Estimation of prediction error curves

The function \texttt{pec} provides several estimates of the prediction error. For right censored data the squared residuals are weighted using inverse probability of censoring weights (IPCW) (Gerds and Schumacher 2006). The weights can be based on the Kaplan-Meier estimate for the survival function corresponding to the censoring times, alternatively on a Cox regression model, or an additive Aalen regression model for the conditional censoring distribution given a linear or non-linear combination of predictor variables, if \( C_i \) is independent of \((T_i, X_i)\) the Kaplan-Meier estimator yields consistent results. For the computation of the prediction error the true status is replaced by the observed status defined as \( \tilde{Y}_i(t) = \mathcal{I}(\tilde{T}_i > t) \). The weights are estimated by

\[
\hat{W}_i(t) = \frac{(1 - \tilde{Y}_i(t))\Delta_i}{\hat{G}(T_i - |X_i)} + \frac{\tilde{Y}_i(t)}{\hat{G}(t|X_i)}
\]

where \( \hat{G}(t|x) \approx P(C_i > t|X_i = x) \) is an estimate of the conditional censoring distribution as noted above.

If an independent test data set \( \tilde{D}_M \) is available the prediction errors are estimated as

\[
\hat{BS}(t, \hat{S}) = \frac{1}{M} \sum_{i \in \tilde{D}_M} \hat{W}_i(t)(\tilde{Y}_i(t) - \hat{S}(t|X_i))^2,
\]

where \( M \) is the number of subjects in \( \tilde{D}_M \) and \( \hat{S} \) is based on a training data.

Furthermore, several methods are implemented to deal with over-fitting in situations where only one data set is available for building the prediction models and for the estimation of prediction performance (Gerds and Schumacher 2007). Optionally the function computes one or all of the following estimates:

- \texttt{AppErr}: apparent or re-substitution estimate.
- \texttt{BootCvErr}: bootstrap cross-validation estimate.
- \texttt{crossvalErr}: k-fold cross-validation estimate.
- \texttt{loocvErr}: leave-one-out cross-validation estimate.
- \texttt{NoInfErr}: no information estimate.
- \texttt{Boot632Err}: Efron’s .632 estimate.
- \texttt{Boot632plusErr}: Efron & Tibshirani’s .632+ estimate.
The apparent estimate of the prediction error re-substitutes the data of the $N$ subjects, $D_N$, that were used to build the models as follows:

$$\text{AppErr}(t, \hat{S}) = \frac{1}{N} \sum_{i \in D_N} \hat{W}_i(t)(\hat{Y}_i(t) - \hat{S}(t|X_i))^2.$$ 

Here $\hat{W}_i$ is estimated using the data of all subjects.

The bootstrap cross-validation approach splits the data $D_N$ into many training samples $D_b$ and corresponding test samples $D_N \setminus D_b$ ($b = 1, \ldots, B$). For one point in time the bootstrap cross-validation estimate of prediction error is estimated by averaging over a test sample and models $\hat{S}_b$ which is based on the training sample:

$$\text{BootCvErr}(t, \hat{S}) = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{M_b} \sum_{i \in D_N \setminus D_b} \hat{W}_i(t)(\hat{Y}_i(t) - \hat{S}_b(t|X_i))^2.$$ 

Here $M_b$ is the number of subjects in $D_N \setminus D_b$. The conditional censoring distribution is by default estimated with all subjects even when bootstrap cross-validation is used. However, the new version of pec has an option that allows the conditional censoring distribution to be estimated separately in each test sample.

We compare the rival modeling strategies at one point in time with the bootstrap .632+ estimate of the prediction error. This estimate is a weighted combination of the apparent estimate, the bootstrap cross-validation estimate and the no information estimate (Efron and Tibshirani 1997; Gerds and Schumacher 2007). The latter permutes the status indicator of the subjects:

$$\text{NoInfErr}(t, \hat{S}) = \frac{1}{N^2} \sum_{j \in D_N} \sum_{i \in D_N} \hat{W}_i(t)(\hat{Y}_j(t) - \hat{S}(t|X_i))^2.$$ 

The bootstrap .632+ estimate of the prediction error is then computed as

$$B632plusErr <- lapply(1:NF,function(f){
  Err1 <- pmin(BootstrapCrossValErr[[f]],NoInfErr[[f]])
  overfit <- (Err1 - AppErr[[f]]) / (NoInfErr[[f]] - AppErr[[f]])
  overfit[!(Err1>AppErr[[f]])] <- 0
  w <- .632 / (1 - .368 * overfit)
  B632plusErr <- (1-w) * AppErr[[f]] + w * Err1
  B632plusErr
})$$

where NF is the number of modeling strategies.

**Remark 1** For data-adaptive modeling strategies one would expect that models selected in different bootstrap samples are different. For example for the algorithm described in Section 2.2, one could select a Cox regression model containing two predictor variables in one bootstrap sample and a model with three predictor variables in another bootstrap sample. Such model uncertainty has been observed by Austin and Tu (2004). Similarly for the random forest approaches the trees will differ across bootstrap samples.
The prediction error curves can be summarized with the integrated Brier score defined as

\[ \text{IBS}(\text{predErr}, \tau) = \frac{1}{\tau} \int_0^\tau \text{predErr}(u, \hat{S}) du, \]

where \( \text{predErr} \) is the chosen method for estimating predictive performance and \( \tau > 0 \) is any value smaller than the minimum of the maximum times for which estimated prediction errors can be evaluated in each bootstrap sample.

**Example (continued)** We fit a pec object with the three rival prediction models `fitcox`, `fitrsf`, and `fitcforest` described in Section 3. The models are passed to pec as a list. In order to obtain all estimates of the prediction error curves, we choose `method=Boot632plus`. We use again the complete case data of the COST study with \( N=518 \) patients. We set \( B=1000 \) and use random subsampling to find training sets of size \( M=350 \), and correspondingly test sets of size \( N-M=168 \). Note that with this option the function pec computes and delivers in the output in addition to the .632+ estimate, the apparent estimate, the bootstrap cross-validation estimate, and the no information estimate of the prediction error curves.

The IPCW, \( W_i \), are estimated using formula (3) dependent on two arguments in pec: the argument `cens.model` specifies the model class, and the right hand side of the argument `formula` specifies the predictor variables that should enter the censoring model. In our example there are few censored observations, and we use the Kaplan-Meier estimator for the censoring distribution. The default option is `cens.model="cox"` that reduces to Kaplan-Meier when we not specify any predictor variables. The left hand side of the argument of `formula` (either a `Surv` object or a `Hist` object) is used to identify the survival response.

The argument `keep.index=TRUE` controls whether or not to keep the indices of the bootstrap samples and `keep.matrix=TRUE` controls whether or not to keep for each model all the estimates of the prediction error curves obtained in the B cross-validation steps.

R> extends <- function(...)TRUE
R> set.seed(13)
R> fitpec <- pec(list("selectcox"=fitcox,"rsf"=fitrsf,"cforest"=fitcforest),
  + data=cost,
  + formula=Surv(time,status)~1,
  + method="Boot632plus",
  + B=1000,
  + M=350,
  + keep.index=TRUE,
  + keep.matrix=TRUE)

Prediction error curves

    Prediction models
1  KaplanMeier
2  selectcox
3    rsf
4   cforest
Evaluating random forests for survival analysis using prediction error curves

IPCW: marginal model
Method for estimating the prediction error:

Bootstrap cross-validation

Type: subsampling
350 out of 518

No. bootstrap samples: 1000
Sample size: 518

Cumulative prediction error, aka Integrated Brier score (IBS)
aka Cumulative rank probability score

Range of integration: 0 and 4213:

IBS[0;4213]
KaplanMeier 0.201
selectcox 0.169
rsf 0.160
cforest 0.171

The print function shows information of the three rival modeling strategies and of the Kaplan-Meier model, a null model added by default.
The integrated Brier scores between 0 and 4213 days for the bootstrap .632+ estimates of the prediction error are lowest for random survival forest. The selected Cox regression model and the conditional inference forest have approximately the same value. All three models perform substantially better than Kaplan-Meier.
The following command plots prediction error curves estimated with the bootstrap .632+ method for all four models in one graph (Figure 2):

R> plot(fitpec, predErr="Boot632plusErr",xlim=c(0,10*365.25),
+       axis1.at=seq(0,10*365.25, 2*365.25), axis1.label=seq(0,10,2))

[Figure 2 about here.]

All curves start at time 0 where all subjects are alive and all predictions equal to 1. The prediction error curve of the benchmark Kaplan-Meier model reaches its maximum value, at the median survival time of 4.9 years. This value is 0.25 for prediction errors estimated with the apparent method. For the bootstrap .632+ estimator, random survival forest clearly outperforms the other strategies. However, the bootstrap cross-validation estimates of the prediction error curves of all three strategies are close to each other (Figure 3) showing that at a sample size of M=350 there is no indication that random survival forest outperforms the other strategies. Figure 3 was produced using the following code:
The bootstrap .632+ estimate is a combination of the apparent estimate, the bootstrap cross-validation estimate, and the no information estimate, so the difference in the prediction error curves for the bootstrap .632+ estimates compared to the bootstrap cross-validation estimates rely on one of the other estimates. To observe how the different estimates behave for each of the three modeling strategies we plot in Figure 4 the apparent-, the bootstrap-, and the no information estimates of the prediction error together with the bootstrap .632+ estimate and 100 prediction error curves obtained during the cross-validation procedure bootstrap. These latter curves were extracted via the argument keep.matrix=TRUE in pec. The following code produces these figures:

```r
R> par(mfrow=c(3,1))
R> lapply(2:4,function(x){
+   plot(fitpec,
+     predErr="Boot632plusErr",
+     models=x,
+     xlim=c(0,10*365.25),
+     axis1.at=seq(0,10*365.25, 2*365.25),
+     axis1.label=seq(0,10,2),
+     special=TRUE,
+     special.addprederr=c("AppErr", "BootCvErr","NoInfErr"))
+ })
```

For both random forest approaches the apparent estimate of the prediction error curves are much lower than the bootstrap cross-validation estimate of the curves.

5. Discussion

We have illustrated how to extend the pec package to two random forest approaches: random survival forest, and conditional inference forest. For illustration, we looked in depth at the COST data, a low-dimensional scenario that is common in medical data settings. We compared three rival modeling strategies (random forests and "automated" Cox regression). We found that using the bootstrap cross-validation estimate of prediction error yielded comparable performances across all methods, however when using the .632+ bootstrap estimate, we found that random survival forests was best. Also, despite the similarity of the overall bootstrap cross-validation performance, we found that the three modeling strategies can predict rather differently at individual predictor values.

An advantage of the pec function is that the prediction error estimates are computed for all rival modeling strategies simultaneously, such that exactly the same bootstrap samples are
used for training of the models. There are several other R-packages for comparing prediction models in survival analysis. The packages `survcomp` (Haibe-Kains, Sotiriou, and Bontempi 2009) and `ipred` (Peters and Hothorn 2008) also estimate the expected Brier score. However, `survcomp` (Haibe-Kains et al. 2009) is restricted to Cox regression models and Kaplan-Meier models. The `ipred` (Peters and Hothorn 2008) package also uses inverse probability of censoring weights, however it does not allow modelling of the censoring distribution dependent on predictor variables. The package `peperr` (Porzelius and Binder 2010) is designed for high-dimensional data analysis and uses parallel computing to efficiently obtain estimates of the prediction error.

Other measures than the Brier score can be used to quantify the predictive performance of a prediction model. The packages `rms` Harrell Jr (2009) and `survcomp` can be used to estimate the concordance index, and the package `survivalROC` (Heagerty and Paramita Saha 2006) offer functions to estimate time-dependent receiver operating curves.

There are other extensions of the `pec` package under construction. One is the van de Wiel test (Van de Wiel, Berkhof, and van Wieringen 2009) for pairwise testing the difference of the prediction error curves for rival modeling strategies at selected time points. Another is an extension for compare modeling strategies in a competing risks situation.

**Acknowledgments**

We owe thanks to Tom Skyhøj Olsen, MD, PhD, for the access to the data from the Copenhagen Stroke Study and to Torsten Hothorn and Axel Benner for valuable information regarding `cforest`.

**References**


**Affiliation:**

Ulla B Mogensen  
Department of Biostatistics  
University of Copenhagen  
Øster Farimagsgade 5, entr. B  
DK-1014 Copenhagen, Denmark  
E-mail: u.b.mogensen@biostat.ku.dk
Evaluating random forests for survival analysis using prediction error curves
List of Figures

1  Predicted survival curves for \texttt{newData[1,]} left panel, \texttt{newData[2,]} middle panel, and \texttt{newData[3,]} right panel. Both random forest approaches used 1000 trees. . 18

2  The bootstrap .632+ estimates of the prediction error based on 1000 bootstrap samples. Both random forest approaches are based on 1000 trees per bootstrap sample. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3  The bootstrap cross-validation estimates of the prediction error based on 1000 bootstrap samples. Both random forest approaches are based on 1000 trees per bootstrap sample. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

4  Each of the three panels shows four different estimates of the prediction error together with a cloud of 100 bootstrap cross-validation curves (grey lines). Both random forest approaches are based on 1000 trees per bootstrap sample. . 21
Figure 1: Predicted survival curves for `newData[1,]` left panel, `newData[2,]` middle panel, and `newData[3,]` right panel. Both random forest approaches used 1000 trees.
Figure 2: The bootstrap .632+ estimates of the prediction error based on 1000 bootstrap samples. Both random forest approaches are based on 1000 trees per bootstrap sample.
Figure 3: The bootstrap cross-validation estimates of the prediction error based on 1000 bootstrap samples. Both random forest approaches are based on 1000 trees per bootstrap sample.
Figure 4: Each of the three panels shows four different estimates of the prediction error together with a cloud of 100 bootstrap cross-validation curves (grey lines). Both random forest approaches are based on 1000 trees per bootstrap sample.
List of Tables

1. The 13 predictor variables of the COST study. Shown are the count (percentage) of patients with factor level "yes" and the minimum and maximum values for continuous predictor variables stratified by gender.  

2. Predicted survival (in %) for newData at 10 years based on the selected Cox regression model (selectCox) and two random forest models: random survival forest (rsf) and conditional inference forest (cforest); both based on 1000 trees.
<table>
<thead>
<tr>
<th>Predictor variables</th>
<th>Coding/Range</th>
<th>Variable name</th>
<th>Female (n=555)</th>
<th>Male (n=438)</th>
<th>Total (n=993)</th>
<th>Missing (n=475)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Factors:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyper tension yes/no</td>
<td>yes/no</td>
<td>hypTen</td>
<td>172 (33.73%)</td>
<td>134 (31.83%)</td>
<td>306 (32.87%)</td>
<td>62 (6.24%)</td>
</tr>
<tr>
<td>Myocardial infarction yes/no</td>
<td>yes/no</td>
<td>myoInf</td>
<td>102 (20.28%)</td>
<td>87 (21.01%)</td>
<td>189 (20.61%)</td>
<td>76 (7.65%)</td>
</tr>
<tr>
<td>Previous stroke yes/no</td>
<td>yes/no</td>
<td>prevStroke</td>
<td>101 (19.46%)</td>
<td>94 (22.17%)</td>
<td>195 (20.68%)</td>
<td>50 (5.04%)</td>
</tr>
<tr>
<td>Other disabling disease yes/no</td>
<td>yes/no</td>
<td>othDisease</td>
<td>130 (24.76%)</td>
<td>75 (17.56%)</td>
<td>205 (21.53%)</td>
<td>41 (4.13%)</td>
</tr>
<tr>
<td>Alcohol intake yes/no</td>
<td>yes/no</td>
<td>alcohol</td>
<td>74 (16.55%)</td>
<td>187 (48.83%)</td>
<td>261 (31.45%)</td>
<td>163 (16.41%)</td>
</tr>
<tr>
<td>Diabetes yes/no</td>
<td>yes/no</td>
<td>diabetes</td>
<td>72 (13.95%)</td>
<td>76 (17.8%)</td>
<td>148 (15.69%)</td>
<td>50 (5.04%)</td>
</tr>
<tr>
<td>Smoking status yes/no</td>
<td>yes/no</td>
<td>smoke</td>
<td>161 (36.51%)</td>
<td>203 (53.7%)</td>
<td>364 (44.44%)</td>
<td>174 (17.52%)</td>
</tr>
<tr>
<td>Hemorrhage yes/no</td>
<td>yes/no</td>
<td>hemor</td>
<td>109 (19.96%)</td>
<td>53 (12.16%)</td>
<td>162 (16.5%)</td>
<td>11 (1.11%)</td>
</tr>
<tr>
<td>Intermittent claudication yes/no</td>
<td>yes/no</td>
<td>intClau</td>
<td>42 (9.5%)</td>
<td>19 (5.23%)</td>
<td>61 (7.58%)</td>
<td>188 (18.93%)</td>
</tr>
<tr>
<td><strong>Continuous:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>min/max</td>
<td>age</td>
<td>25/98</td>
<td>28/93</td>
<td>25/98</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Scandinavian stroke score</td>
<td>min/max</td>
<td>strokeScore</td>
<td>0/58</td>
<td>0/58</td>
<td>0/58</td>
<td>7 (0.7%)</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>min/max</td>
<td>cholest</td>
<td>1.7/11.6</td>
<td>1.5/10.5</td>
<td>1.5/11.6</td>
<td>233 (23.46%)</td>
</tr>
</tbody>
</table>

Table 1: The 13 predictor variables of the COST study. Shown are the count (percentage) of patients with factor level "yes" and the minimum and maximum values for continuous predictor variables stratified by gender.
<table>
<thead>
<tr>
<th>Patient Id</th>
<th>Age</th>
<th>selectCox</th>
<th>rsf</th>
<th>cforest</th>
</tr>
</thead>
<tbody>
<tr>
<td>newData 1</td>
<td>28</td>
<td>86.05</td>
<td>54.2</td>
<td>47.73</td>
</tr>
<tr>
<td>newData 2</td>
<td>74</td>
<td>24.17</td>
<td>34.31</td>
<td>20.66</td>
</tr>
<tr>
<td>newData 3</td>
<td>95</td>
<td>1.91</td>
<td>13.56</td>
<td>9.74</td>
</tr>
</tbody>
</table>

Table 2: Predicted survival (in %) for **newData** at 10 years based on the selected Cox regression model (**selectCox**) and two random forest models: random survival forest (**rsf**) and conditional inference forest (**cforest**); both based on 1000 trees.
08/01 Siersma, V. & Kreiner, S. Monte Carlo Evaluation of Model Search in Graphical Models for Ordinal Data.

08/02 Andersen, P.K. & Pohar Perme, M. Inference for outcome probabilities in multi-state models.

08/03 Scheike, T.H. & Zhang, M. Flexible competing risks regression modelling and goodness-of-fit.


08/05 Gerds, T.A., Qvist, V., Strub, J.R. & Keiding, N. Survival analysis in dental research: The typical observational patterns.

08/06 Carstensen. B. Limits of agreement: How to use the regression of differences on averages.


08/08 Martinussen, T. & Scheike, T.H.. Covariate selection for the semiparametric additive risk model.


08/10 Martinussen, T. & Scheike, T.H. The Aalen additive hazards model with high-dimensional regressors.


08/12 Andersen, P.K., Pohar Perme, M. Pseudo-observations in survival analysis.
10/6 Andersen, P.K. & Keiding, N. Interpretability and importance of functionals in competing risks and multi-state models.

10/7 Gerds, T.A., Kattan, M.W., Schumacher, M. & Yu, C. Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring.