Extended inference with lasso and elastic-net regularized Cox and generalized linear models

Martin Sill*, Thomas Hielscher, Natalia Becker, Manuela Zucknick

Division of Biostatistics, German Cancer Research Center, Im Neuenheimer Feld 280, 69120 Heidelberg, Germany

Abstract

We have developed the R package `c060` with the aim of improving R software functionality for high-dimensional risk prediction modelling, e.g. for prognostic modelling of survival data using high-throughput genomic data.

Penalized regression models provide a statistically appealing way of building risk prediction models from high-dimensional data. The popular CRAN package `glmnet` package implements an efficient algorithm for fitting penalized Cox and generalized linear models. However, in practical applications the data analysis will typically not stop at the point where the model has been fitted. One is for example often interested in the stability of selected features and in assessing the prediction performance of a model and we provide functions to deal with both of these tasks. Our R functions are computationally efficient and offer the possibility of speeding up computing time through parallel computing. Another feature which can drastically reduce computing time is an efficient interval-search algorithm, which we have implemented for selecting the optimal parameter combination for elastic net penalties.

These functions have been useful in our daily work at the German Cancer Research Center where prognostic modelling of patient survival data is of particular interest. Although we focus on a survival data application of penalized Cox models in this article, the functions in our R package are applicable to all types of regression models implemented in the `glmnet` package.

Keywords: penalized log-likelihood method, stability selection, interval search, prediction error, R software

*Corresponding author: m.sill@dkfz-heidelberg.de

1. Introduction

Penalized regression models provide a statistically appealing method to build prediction models from high-dimensional data sources, where it is the aim to simultaneously select features and to fit the model (Fan and Lv, 2010; Benner et al., 2010). Since the introduction of the lasso for linear regression models (Tibshirani, 1996), the methodology has been extended to generalized linear regression models and time-to-event endpoints (Tibshirani, 1997) among others. In addition to the well-known L_1- (lasso) and L_2-norm (ridge) penalty functions, various other penalties have been proposed in recent years to select features and/or estimate their effects. In particular, we will use the elastic net penalty function (Zou and Hastie, 2005), which is a linear combination of the L_1- and L_2-norms.

With ever increasing data, the properties of the algorithm used for fitting the model have become almost as important as the statistical model itself. In 2010, Friedman et al. proposed a coordinate descent algorithm (Friedman et al., 2010) for generalized linear regression models, which has since then been extended to penalized Cox proportional hazards (PH) regression models (Simon et al., 2011). Due to its efficiency this algorithm is considered one of the state-of-the-art approaches to estimate penalized regression models with lasso, ridge or elastic net penalty terms, especially in high-dimensional data scenarios.

This algorithm has been implemented in R (R Development Core Team, 2011) in the `glmnet` package. The package provides functions to tune and fit regression models, plot the results, and make predictions. However, in practical applications, where often an independent validation data set is lacking, some additional features and routines are desirable as part of a complete data analysis. We have assembled some functions that enhance the existing functionality of the `glmnet` package or allow to use it within the framework of other existing R packages. These functions have been useful in our daily work at the German Cancer Research Center where prognostic modelling of patient survival data is of particular interest. Therefore, for illustration purposes we focus on penalized Cox PH regression models in this article. But the R functions are applicable to all types of regression models implemented in the `glmnet` package.

Computational efficiency is an important requirement of the software to make applications feasible for real-life data analysis tasks in fields such as molecular oncology, where one aims to develop sparse risk prediction models based on very large numbers of molecular features measured with high-throughput technologies such as microarrays or next-generation sequencing.

Therefore, we provide functionality to speed up computations, in particular through parallel computing.

We provide R functions to perform stability selection (Meinshausen and Bühlmann, 2010) in a computationally efficient way using `glmnet` which allows to select the most stable features at a given error level. We have also implemented an approach to select the optimal parameter combination (α, λ) for elastic net penalties using an interval-search algorithm (Froehlich and Zell, 2005) which is often faster and more accurate than a standard grid search (**?**). Another very useful addition for real-life applications of `glmnet` for building risk-prediction models is the provision of wrapper functions to allow the computation of resampling-based prediction errors within the framework of the R package `peperr` (Porzelius et al., 2009). The `peperr` package makes it computationally feasible to assess the predictive accuracy of a penalized regression model via resampling methods even for very large-scale applications by employing parallel computing. We also provide the possibility to speed up stability selection by parallel computing using the functionalities of the R base package `parallel`.

The software is available as R package `c060` on R-forge (URL `http://c060.r-forge.r-project.org`).

2. Data application

Throughout this article we use the gene expression data set of cytogenetically normal acute myeloid leukemia (AML) patients by Metzeler et al. (2008) and corresponding clinical data in order to illustrate a typical application of penalized Cox PH regression models, where the aim is to develop a prognostic model for patient survival while at the same time identifying the most influential gene expression features. To simulate the typical situation that only one data set is available for model training and evaluation, we only use the data set that was used as validation data in the original publication by Metzeler et al. (2008). The data can be accessed from the Gene Expression Omnibus (GEO) data repository (`http://www.ncbi.nlm.nih.gov/geo`) by the National Center for Biotechnology Information (NCBI). We find the Metzeler *et al.* data set under GEO accession number GSE12417.

The data set contains gene expression data for 79 patient samples measured with Affymetrix HG-U133 Plus 2.0 microarrays. The median survival time of these 79 patients was 17.6 months with a censoring rate of 40%.

3. Methods and algorithms

3.1. Penalized generalized linear models and Cox models

An efficient implementation for fitting generalized linear models and Cox proportional hazards models with regularization by the lasso or elastic net penalty terms is provided by the R package `glmnet` (Friedman et al., 2010; Simon et al., 2011). This implementation uses a coordinate descent algorithm for fitting the models for a specified penalty parameter value λ. The computation of an entire regularization path across a range of values $\Lambda = \{\lambda_1, \lambda_2 ..., \lambda_K\}$ with `glmnet` is generally very fast, because previously computed solutions for $\{\lambda_1, ..., \lambda_{k-1}\}$ are used as 'hot' starting values for the computation of λ_k. This implies that it is often more efficient to compute the models for the entire regularization path rather than just individual models. We use this feature of `glmnet` in all of our implemented algorithms to make most use of the computational speed of `glmnet`.

Models are fitted by maximizing the penalized log-likelihood function for generalized linear models and the penalized partial log-likelihood for Cox models. The penalized (partial) log-likelihood function is given by

$$l_n(\beta) - \sum_{j=1}^{p} p_\lambda(|\beta_j|) \tag{1}$$

where $l_n(\beta)$ denotes the (partial) log-likelihood given n observations. The dimension of the parameter vector β is p and $p_\lambda(|\cdot|)$ is the penalty function with tuning parameter λ.

Cross-validation can be performed to decide which model, i.e. which penalty parameter values, to choose by using the negative cross-validated (partial) log-likelihood as the loss function. Actually, within the `glmnet` package, the (partial) log-likelihood deviance is used as the loss function rather than the log-likelihood function itself. The deviance is equal to âĹŠ2 times the log-likelihood ratio of the model of interest compared to the saturated model, which has one free parameter per observation. Obviously, both versions will result in the same optimization result.

3.2. L_2-penalized Cox regression

Penalized maximum likelihood estimation in Cox regression with the ridge penalty

$$p_\lambda(|\beta_j|) = \lambda \beta_j^2 \tag{2}$$

was introduced by Verweij and van Houwelingen (1994). The ridge penalty results in parameter estimates that are biased towards zero, but does not set

values to exactly zero, and hence does not perform variable selection. On the other hand, it has been found to produce models with good prediction performance in high-dimensional genomic applications (e.g. Bøvelstad et al., 2007), in particular if predictors are highly correlated.

3.3. L_1-penalized Cox regression

Tibshirani (1997) proposed to use an L_1-penalized Cox model with

$$p_\lambda(|\beta_j|) = \lambda|\beta_j| \tag{3}$$

and described a technique, called the lasso for "least absolute shrinkage and selection operator", for parameter estimation. The L_1-penalty has the advantage over the L_2-penalty of shrinking some of the coefficients to zero, i.e. it performs automatic variable selection.

3.4. The elastic net

Zou and Hastie (2005) introduced the elastic net, which employs a combination of the L_1- and L_2-penalty

$$p_{\lambda_1,\lambda_2}(|\beta_j|) = \lambda_1|\beta_j| + \lambda_2\beta_j^2. \tag{4}$$

Zou and Hastie (2005) rescale the initial solutions from the optimization of the doubly-penalized log-likelihood function by the factor $1+\lambda_2$, in order to reduce the effect of the double shrinkage. Like lasso the elastic net performs automatic variable selection by setting some coefficient estimates to zero. But the additional L_2-penalty term distributes the weight to more variables, such that the elastic net tends to select more variables than the lasso. This is especially the case in situations with high correlation, where the lasso would select only one variable of a set of highly correlated variables, while the ridge penalty would give them equal weight.

Throughout this manuscript we use an alternative parametrization of the elastic net penalty function equivalently to the formulation used in the `glmnet` package:

$$p_{\alpha,\lambda}(|\beta_j|) = \lambda \times (\alpha|\beta_j| + (1-\alpha)\frac{1}{2}|\beta_j|^2). \tag{5}$$

Here, $\alpha \in (0,1]$ determines the influence of the L_1 penalty relative to the L_2 penalty. Small α values will result in models with many variables, getting closer to the non-sparse ridge solution as α tends to zero.

3.4.1. The interval-search algorithm to select the optimal elastic net parameter combination

The elastic net penalty function contrains two tuning parameters which are data-dependent and cannot be set equal to some apriori values. The challenge is to find such a setting of tuning parameters (α, λ), for which the k-fold cross validation error of the model is minimal.

Fixed grid. Tuning parameters are usually determined by a grid search. The grid search method calculates a target value, e.g. the misclassification rate, at each point over a fixed grid of parameter values. This method may offer some protection against local minima but it is not very efficient. The density of the grid plays a critical role in finding global optima. By increasing the density of the grid, the computation cost increases rapidly with no guaranty of finding global optima. Ffurthermore, the choice of the fixed grid is highly arbitrary and do not protect from stucking in local optima.

Interval search. Instead of using a fixed grid, an interval search approach can be applied. Froehlich and Zell (2005) proposed an efficient algorithm for finding a global optimum on the tuning parameter space called Efficient Parameter Selection via Global Optimization (EPSGO). The main idea of the algorithm is to treat the finding optimal tuning parameter problem as a global optimization problem. For that purpose one learns a Gaussian Process model of the error surface in parameter space and samples systematically at points for where a specific parameter namely expected improvement criterion reaches the maximum.

The algorithm learns the Gaussian Process (GP) model from the points in the parameter space which have been already visited. An immense improvement in the training time for the GP compared to that of the classification and regression tasks was be observed (Froehlich and Zell, 2005). Since the number of training points for the GP and hence the number of evaluations of the error surface of the Cox model mainly depends on the dimensionality of the parameter space, which is very small compared to the number of training points for the Cox model. Finally, new points in the parameter space are sampled by using the expected improvement criterion as described by **?**, which avoids sticking in local minima. The EPSGO algorithm stops when the global optimum is achieved.

3.5. Stability selection

In contrast to the previously described methods that are typically used to find sparse but also well predicting models, the stability selection proposed

by Meinshausen and Bühlmann (2010) aims to find stable features which show strong association with the outcome. The stability selection is a general approach that combines variable selection methods such as L_1 penalized Cox models with resampling. By applying the corresponding variable selection method to subsamples that were drawn without replacement, selection probabilities for each feature can be estimated. These selection probabilities are used to define a set of stable features. Meinshausen and Bühlmann (2010) provide a theoretical framework for controlling Type I error rates of falsely assigning features to the estimated set of stable features. The selection probability of each feature along the regularization path, e.g. the range of possible penalization parameters $\Lambda = \{\lambda_1, \lambda_2 ..., \lambda_k\}$, is called stability path. Given an arbitrary threshold $\pi_{thr} \in (0.5, 1)$ and the set of penalization parameters Λ, the set of stable features estimated with the stability selection is:

$$\hat{S}_{\beta}^{stable} = \left\{ i : \max_{\lambda_j \in \Lambda} \hat{\Pi}_i^{\lambda_j} \geq \pi_{thr} \right\}, \tag{6}$$

where $\hat{\Pi}_i^{\lambda_j}$ denotes the estimated selection probability of the ith feature at the jth λ. Then according to Theorem 1 in Meinshausen and Bühlmann (2010), the expected number of falsely selected features $E(V)$ will be bounded by:

$$E(V) \leq \frac{1}{(2\pi_{thr} - 1)} \frac{q_{\Lambda}^2}{p}, \tag{7}$$

where q_{Λ} is the average of the number of non-zero coefficients w.r.t. to the drawn subsamples. Interpreting Equation 7 the expected number of falsely selected features decreases by either reducing the average number of selected features q_{Λ} or by increasing the threshold π_{thr}. Suppose that π_{thr} is fixed, then the stability selection controls $E(V)$ as long as the average number of selected features is less then e_{Λ}. This is an upper bound

$$e_{\Lambda} = \sqrt{E(V) p (2\pi_{thr} - 1)}, \tag{8}$$

which can be controlled by reducing the length of the regularization path Λ. In multiple testing the expected number of falsely selected variables is also known as the per-family error rate (PFER) and if divided by the total number of features p it will become the per-comparison error rate (PCER) Dudoit et al. (2003). The stability selection allows to control these Type I error rates. For instance, suppose the threshold $\pi_{thr} = 0.8$ is fixed, then choosing Λ such that $q_{\Lambda} \leq \sqrt{0.6p}$ will control $E(V) = 1$. Moreover, by choosing Λ so that $q_{\Lambda} \leq \sqrt{0.6p\alpha}$ will control the family wise error rate (FWER)

at level α, $P(|V| > 0) \leq \alpha$.
According to Friedman et al. (2010) the coordinate descent algorithm implemented in the `glmnet` is most efficient regarding the computational time, when used to calculate a whole regularization path. To utilize this property our algorithm calculates the stability path by first generating subsets and then calculating for each subset the regularization path using the coordinate descent algorithm. The resulting regularization paths are then merged to form the stability path. Furthermore, since the calculations of the regularization paths for each subset are independent of each other, the algorithm can easily be parallelized using the package `parallel`.

3.6. Prediction error curves for survival models

The time-dependent Brier Score (Graf et al., 1999) can be used to assess and compare the prediction accuracy of prognostic model. The Brier Score at time point t is a weighted mean squared error between predicted survival probability and observed survival status. Weighting depends on the estimated censoring distribution to account for the observations under risk or more general estimates thereof (Gerds and Schumacher, 2006). Computing the error for each time point over the entire follow-up horizon yields a predicton error curve. Prediction errors based on the Kaplan-Meier estimates ignoring any additional covariate information function as reference.

The empirical time-dependent Brier score $BS(t)$ is defined as a function of time $t > 0$ by

$$BS(t) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\hat{S}(t|x_i)^2 I(t_i \leq t \wedge \delta_i = 1)}{\hat{G}(t_i)} + \frac{(1-\hat{S}(t|x_i))^2 I(t_i > t)}{\hat{G}(t)}\right],$$

with individual survival time t_i, censoring indicator δ_i and estimated survival probability $\hat{S}(t|x_i)$ at time t based on the prognostic model given covariate values x_i. $\hat{G}(t)$ denotes the Kaplan-Meier estimate of the censoring distribution which is based on the observations $(t_i; 1-\delta_i)$, I stands for the indicator function.

In case no independent validation data are available, resampling-based prediction error curves are used to adequately assess the accuracy. The .632+ bootstrap estimator (Efron and Tibshirani, 1997), which is a weighted sum of the apparent error and the average out-of-bag bootstrap error, can be used in this context, balancing a too optimistic and a too conservative error estimation.

4. Application and demonstration of software

In the following we will demonstrate the use of the functions provided with the `c060` package in an application to the AML data set by Metzeler et al. (2008). For the sake of convenience we reduce the total number of 54675 gene expression features that have been measured with the Affymetrix HG-U133 Plus 2.0 microarray technology to the top 10000 features with largest variance across all 79 samples. For all computations the data set is stored as an `ExpressionSet` (from Bioconductor package `Biobase`) called `eset`. Gene expression data can be accessed through the call `exprs(eset)` and overall survival data and other patient-specific data (e.g. patient age) are stored within the `phenoData` object `pData(eset)`.

4.1. Starting off: Fitting the lasso-penalized Cox model

Our goal is to develop a prognostic model for patient overall survival based on the gene expression data. The purpose of this modelling exercise is not just to fit a prognostic model that is capable of predicting overall survival rates, but we also want to find out which gene expression features are most relevant for this task. Traditionally, this problem can be solved by variable selection methods and we start our data analysis exercise by fitting the lasso-penalized Cox model, which provides automatic variable selection.

We can apply the `glmnet` function to fit a lasso-penalized Cox model to the AML data set. The function call with default penalty parameter settings will fit the lasso model for 100 λ values with a data-derived range of values:

```
> fit <- glmnet(y=Surv(pData(eset)$os, pData(eset)$os_status),
+               x=t(exprs(eset)), family="cox")
```

In order to determine the optimal lasso penalty parameter value, we perform 10-fold cross-validation using the `cv.glmnet` function. The loss function, i.e. the cross-validated partial log-likelihood deviance, is shown in Figure 1 including upper and lower standard deviations as a function of $\log\lambda$ for the AML data set. The penalty parameter value minimizing the loss function is λ =0.265 ($\log\lambda$ =-1.329) and corresponds to a final lasso model with the following 5 selected features:

```
  203640_at 204419_x_at 222462_s_at   226169_at   233371_at
-0.11339033 -0.01664530  0.27420521  0.04300559 -0.01216429
```

The selected features are highlighted as red lines in the coefficient paths shown in Figure 2. The coefficient path shows the development of the regression coefficient estimates with increasing regularization. While the selected

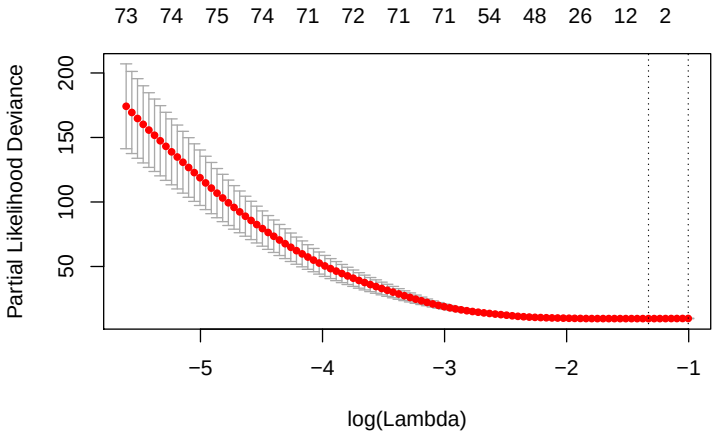

Figure 1: Cross-validated partial log-likelihood deviance, including upper and lower standard deviations, as a function of $\log\lambda$ for the AML data set. The dotted vertical lines indicate the λ values with minimal deviance (left) and with the largest λ value within one standard deviation of the minimal deviance (right).

5 features are the only features selected at the optimal λ value, they do not remain among the features with largest effect sizes when the penalty is reduced and thus more and more coefficients start to enter the model. In fact, for 4 out of the 5 features, the coefficient estimates go back down to zero for small values of $\log\lambda$, indicating that these features get replaced by other gene expression variables in very large models.

4.2. Assessment of prediction performance with resampling-based prediction errors

Once the final prognostic model is selected, we need to assess its prediction accuracy for future patients, frequently also in comparison with established clinico-pathological prognostic markers. In many applications no independent validation data set is available. The same data set need to be used to develop and assess the prognostic model. This is even more problematic for high-dimensional data, where the risk of overfitting is much more present. Resampling-based methods can be used to unbiasedly estimate the predictive accuracy of the prognostic model in this situation. This is also called internal validation or pre-validation.

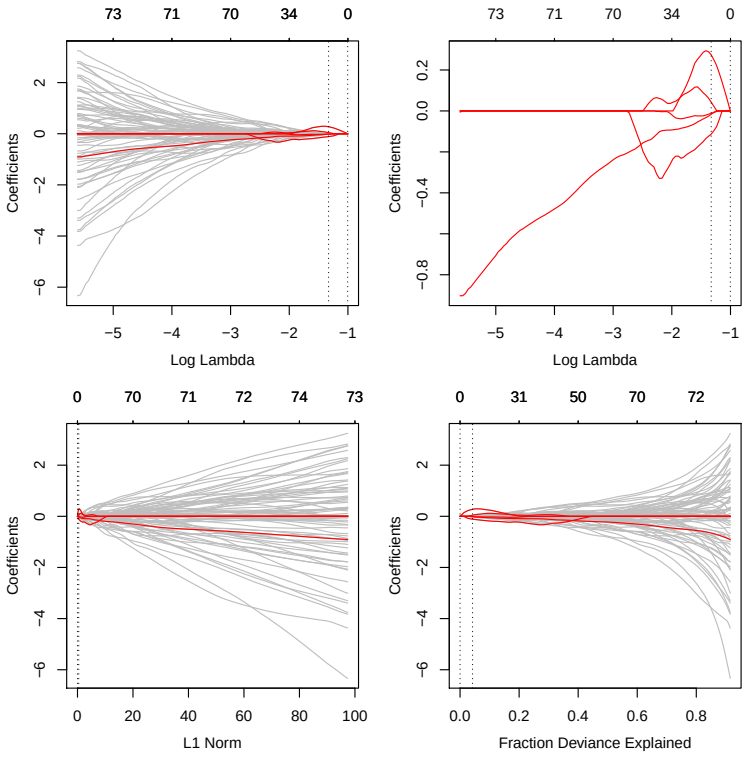

Figure 2: Coefficient paths for lasso-penalized Cox PH regression models applied to the AML data set. The features with highlighted paths have non-zero coefficients in the model with the optimal λ value as determined by ten-fold cross-validation. The dotted vertical have the same meaning as in Figure 1. The top plots show the coefficient path scaled to reflect $\log(\lambda)$ on the x-axis (top left: full path, top right: zoomed in to only show the selected features). The bottom plots show the coefficient paths relative to the L_1-norms of the estimated coefficient vector (left) and to the fraction of the null partial log-likelihood deviance explained (right).

The R package peperr (Porzelius et al., 2009) provides a modular framework for survival and binary endpoints, i.e. prognostic and classification models. Wrapper functions for new or customized prediction model algorithms can be defined and passed to the generic call function peperr. In case of prognostic models, algorithm specific wrapper functions for model fitting, tuning and prediction are required. Wrapper functions for selected machine learning approaches are already implemented.

Prediction accuracy is per default assessed with prediction error curves based on the time-dependent Brier score (Graf et al., 1999). But it is also possible to define and use customized accuracy measures.

We defined additional wrapper functions for the glmnet algorithm for fitting (fit.glmnet) and tuning (complexity.glmnet) the model, and predicting survival probabilities (predictProb.glmnet) based on the fitted model and the estimated baseline hazard from the training data.

We estimate the L_1-penalized Cox PH regression model for overall survival starting with the 10.000 most varying probe sets using glmnet. The .632+ bootstrap estimator is calculated based on subsampling (Binder and Schumacher, 2008) using only 100 bootstrap samples for illustration.

The peperr package is designed for high-dimensional covariates data and allows for various set-ups of parallel computations. Also, additional arguments can be passed directly to the glmnet call by specifing additional arguments for the fitting and/or tuning procedure. Here, we include patient's age as mandatory model variable into the prognostic model, i.e. age is not subject to penalization, and run the calculation on 3 CPUs in parallel using a socket cluster set-up.

```
> obj  <- peperr(response=Surv(pData(eset)$os, pData(eset)$os_status),
+          x=data.frame(eset$age,t(exprs(eset))),
+          fit.fun=fit.glmnet,args.fit=list(standardize=F,family="cox",
+          penalty.factor=rep(0:1,times=c(1,dim(eset)[1]))),
+          complexity=complexity.glmnet,
+          args.complexity=list(standardize=F,nfolds=10,
+          family="cox",penalty.factor=rep(0:1,times=c(1,dim(eset)[1]))),
+          trace=F,RNG="fixed",seed=0815,cpus=3,parallel=T,clustertype="SOCK",
+          load.list=list(functions=c("basesurv")),
+          indices=resample.indices(n=dim(eset)[2],sample.n=100,method="sub632"))
```

Individual bootstrap results can be visualized with the plot.peperr function from the peperr package showing the selected complexity parameters, out-of-bag prediction error curves as well as the prediction error integrated over time, and the predictive partial log-likelihood (PLL) values.

In order to calculate the predictive PLL values again an algorithm specific wrapper (here `PLL.coxnet`) needs to be defined.

In addition, we provide a slightly modified version of the prediction error curves plot function from the `peperr` package which allows to display the number still at risk (`plot.peperr.curves`) as shown in figure 3.

Note, that for classification models, the same wrapper functions for fitting and tuning the model are called. Model performance measures shipped with the `peperr` packages are misclassification rate and Brier score.

We extended functionality of the Brier score (`aggregation.brier`) and misclassification rate (`aggregation.misclass`) calculation for the `glmnet` algorithm, and defined AUC under the ROC curve (`aggregation.auc`) as additional performance measure. For binary responses, the `peperr` package does not quite provide the same modular flexibility as for time-to-event endpoints. The predicted class probability is calculated within the performance/aggregation function by calling the algorithm specific predict function. Whenever a new algorithm is incorporated the aggregation function has to be modified and overwritten accordingly.

```
> plot.peperr.curves(obj, at.risk=T)
```

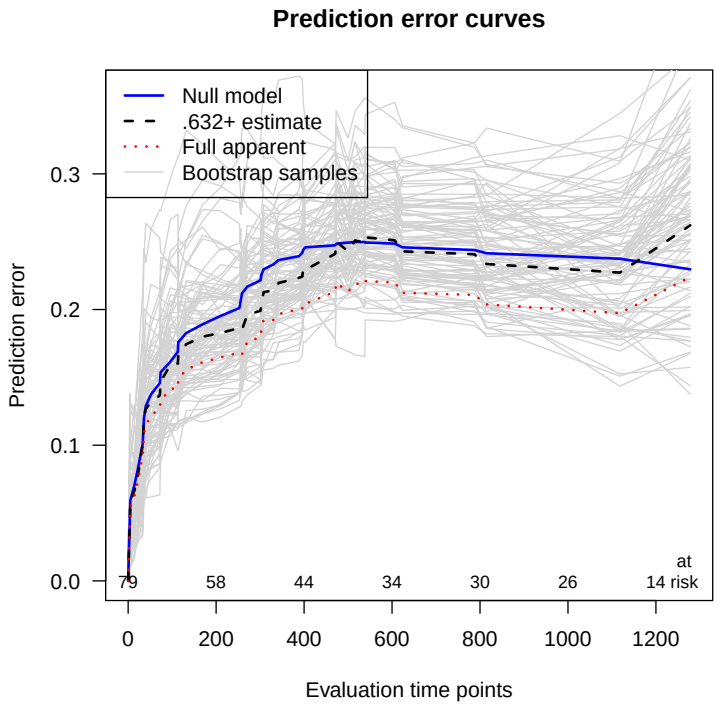

Figure 3: Prediction error curves based on time-dependent Brier score for the lasso-penalized Cox PH regression model applied to the AML data set (evaluation time points reflect days). The gray lines show the prediction error curves of 100 bootstrap samples. The other lines show the prediction error curves of the null model (estimated by the Kaplan-Meier estimator without covariate information), the full apparent error estimates (i.e. the errors as estimated when applying the model to the entire training data set), and the .632+ bootstrap error estimates.

4.3. Stability selection

To identify genes which have a relevant influence on the survival times of the patients in the AML data set and to control a Type I error that the genes identified are truly associated with the survival times, the stability selection Meinshausen and Bühlmann (2010) can be used. To calculate the stability path for the L_1-penalized Cox regression we use the function `stability.path`. The function draws subsets and calculates in parallel the stability path, e.g. the selection probabilities of each feature along the range of possible penalization parameters. For parallelization we use the package `parallel`, which is since R 2.14.0 a base package. On Unix like systems the parallelization is done by forking via the function `mclapply` whereas under Windows systems socket cluster are used. To estimate the stable set of genes given a desired Type I error rate, the function `stability.selection` or `plot.stabpath` can be called. Here, we used the function `plot.stabpath` which will also plot the regularization path for the complete data set along with the stability path, where the stable features are marked in red.

Controlling a FWER of 0.5 the estimated set of stable features comprises a single gene (with $\hat{\Pi}$ >0.6 at λ =0.109) which is:

```
206932_at
     2823
```

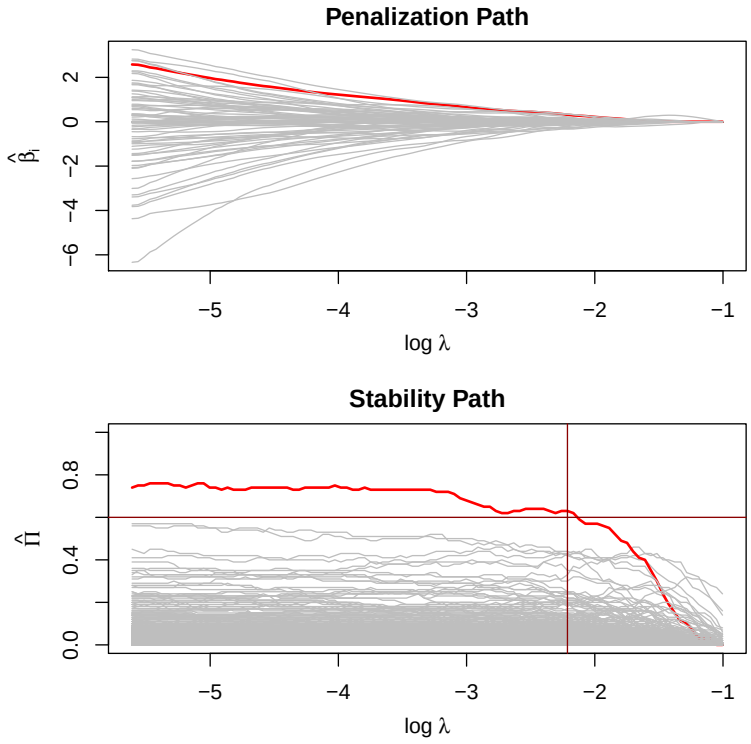

Figure 4: Coefficient and stability paths for lasso penalized Cox PH regression model applied to the AML data set. The feature with highlighted path is the only stable feature found by stability selection with FWER=0.5 and $\hat{\Pi} > 0.6$.

4.4. Elastic net penalized Cox PH regression model

In this section both λ and α are tuned by the interval search.

The task is to find such a setting of tuning parameters (α, λ), for which the 10-fold cross validation error of the model is minimal. Instead of using a fixed grid, an interval search approach is applied.

The parameter space of tuning parameters is defined as follows:

```
      lower upper
alpha     0     1
```

The second tuning parameter $\lambda > 0$ will be found for each given α via regularization path. Thus, the two dimensional parameter space has the form $(0, 1) \times \mathbb{R}_+$. For each given α an optimal lambda is definded as largest value of lambda such that error is within one standard error of the minimum. (parameter type.min = 'lambda.1se').

The first ten 'visited' points (out of 3700)in the parameter space (α, λ) with corresponding cross-validated deviance and number of selected features in each model are presented in figure **??**.

The minimal mean cross-validated deviance over the folds of 9.633 was reached for optimal parameter pair $(\alpha, \lambda) = (0.013, 28.235)$.

The final model selected 0 features:

```
> summary(cofn)
```

```
Length  Class   Mode
     0   NULL   NULL
```

```
> head(sort(cofn))
```

```
NULL
```

```
> tail(sort(cofn))
```

```
NULL
```

A feature selected by the stability algorithm is in the set of selected features

```
> '206932_at' %in% names.cof
```

```
[1] FALSE
```

TODO : 3D/ 2D plots

The selected features are highlighted as red lines in the coefficient paths shown in Figure **??**.

5. Conclusions and outlook

The programming language and statistical computing environment R provides a highly useful framework for statistical data analysis and modelling. It is the dominating statistical software in many areas, for example in molecular biology and medicine, which is largely due to the highly successful Bioconductor project, which provides tools for the analysis and comprehension of high-throughput genomic data. Due to the open-source nature of R and Bioconductor, many very useful software packages have been developed by R users and made available for the entire R community. One example is the `glmnet` package, which implements an efficient state-of-the-art algorithm for fitting penalized Cox and generalized linear models (Friedman et al., 2010; Simon et al., 2011).

We have presented our R package `c060`, which provides extensions to `glmnet` and additional features which are essential for a full data analysis in practical applications, including stability selection, estimation of prediction error (curves) and an efficient interval search algorithm for finding the optimal elastic net tuning parameter combination. These extensions have proved useful in our daily work, in particular for the task of performing prognostic modelling of patient survival data based on high-dimensional molecular biology data.

The `c060` package will be kept updated in the future to keep up with the fast-developing field of penalized regression methodology for variable selection and risk prediction modelling with high-dimensional input data. One example are developments for the estimation of standard errors, confidence intervals and the determination of p-values in high-dimensional regularized regression models, e.g. through subsampling methods similar to the approach taken by Wasserman and Roeder (2009) and Meinshausen and Bühlmann (2010).

References

Benner, A., Zucknick, M., Hielscher, T., Ittrich, C., Mansmann, U., 2010. High-dimensional Cox models: The choice of penalty as part of the model building process. Biometrical Journal 52, 50–69.

Binder, H., Schumacher, M., 2008. Adapting prediction error estimates for biased complexity selection in high-dimensional bootstrap samples. Statistical Applications in Genetics and Molecular Biology 7.

Bøvelstad, H.M.M., Nygård, S., Størvold, H.L.L., Aldrin, M., Borgan, O., Frigessi, A., Lingjærde, O.C.C., 2007. Predicting survival from microarray data - a comparative study. Bioinformatics 23, 2080–2087.

Dudoit, S., Shaffer, J.P., Boldrick, J.C., 2003. Multiple Hypothesis Testing in Microarray Experiments. Statistical Science 18, 71–103.

Efron, B., Tibshirani, R., 1997. Improvements on cross-validation: The. 632+ bootstrap method. Journal of the American Statistical Association , 548–560.

Fan, J., Lv, J., 2010. A selective overview of variable selection in high dimensional feature space. Statistica Sinica 20, 101–148.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33, 1–22.

Froehlich, H., Zell, A., 2005. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization, in: Proceedings of the International Joint Conference of Neural Networks, pp. 1431–1438.

Gerds, T., Schumacher, M., 2006. Consistent estimation of the expected brier score in general survival models with right-censored event times. Biometrical Journal 48, 1029–1040.

Graf, E., Schmoor, C., Sauerbrei, W., Schumacher, M., 1999. Assessment and comparison of prognostic classification schemes for survival data. Statistics in Medicine 18, 2529–2545.

Meinshausen, N., Bühlmann, P., 2010. Stability selection. Journal of the Royal Statistical Society - Series B: Statistical Methodology 72, 417–473.

Metzeler, K., Hummel, M., Bloomfield, C., Spiekermann, K., Braess, J., Sauerland, M.C., Heinecke, A., Radmacher, M., Marcucci, G., Whitman, S., Maharry, K., Paschka, P., Larson, R., Berdel, W., Buchner, T., Wormann, B., Mansmann, U., Hiddemann, W., Bohlander, S., Buske, C., 2008. An 86 probe set gene expression signature predicts survival in cytogenetically normal acute myeloid leukemia. Blood 112(10), 4193–4201.

Porzelius, C., Binder, H., Schumacher, M., 2009. Parallelized prediction error estimation for evaluation of high-dimensional models. Bioinformatics 25, 827–829.

R Development Core Team, 2011. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. ISBN 3-900051-07-0.

Simon, N., Friedman, J., Hastie, T., Tibshirani, R., 2011. Regularization paths for cox's proportional hazards model via coordinate descent. Journal of Statistical Software 39, 1–13.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B: Methodological 58, 267–288.

Tibshirani, R., 1997. The lasso method for variable selection in the Cox model. Statistics in Medicine 16, 385–395.

Verweij, P.J.M., van Houwelingen, H.C., 1994. Penalized likelihood in Cox regression. Statistics in Medicine 13, 2427–2436.

Wasserman, L., Roeder, K., 2009. High dimensional variable selection. The Annals of Statistics 37, 2178–2201.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B 67(2), 301–320.