C060 Vignette

Martin Sill, Thomas Hielscher, Natalia Becker, Manuela Zucknick

1 Introduction

Data from published gene expression studies are often deposited in public data repositories, for example on the Gene Expression Omnibus (GEO) website by the NCBI (National Center for Biotechnology Information): `http://www.ncbi.nlm.nih.gov/geo`. We find the Metzeler *et al.* data under GEO accession number GSE12417.

Here should follow a detailed description of

1. the problem (what do we want to do)
2. the existing methods and R software (what exists in glmnet package and which functions are missing)
3. the data set

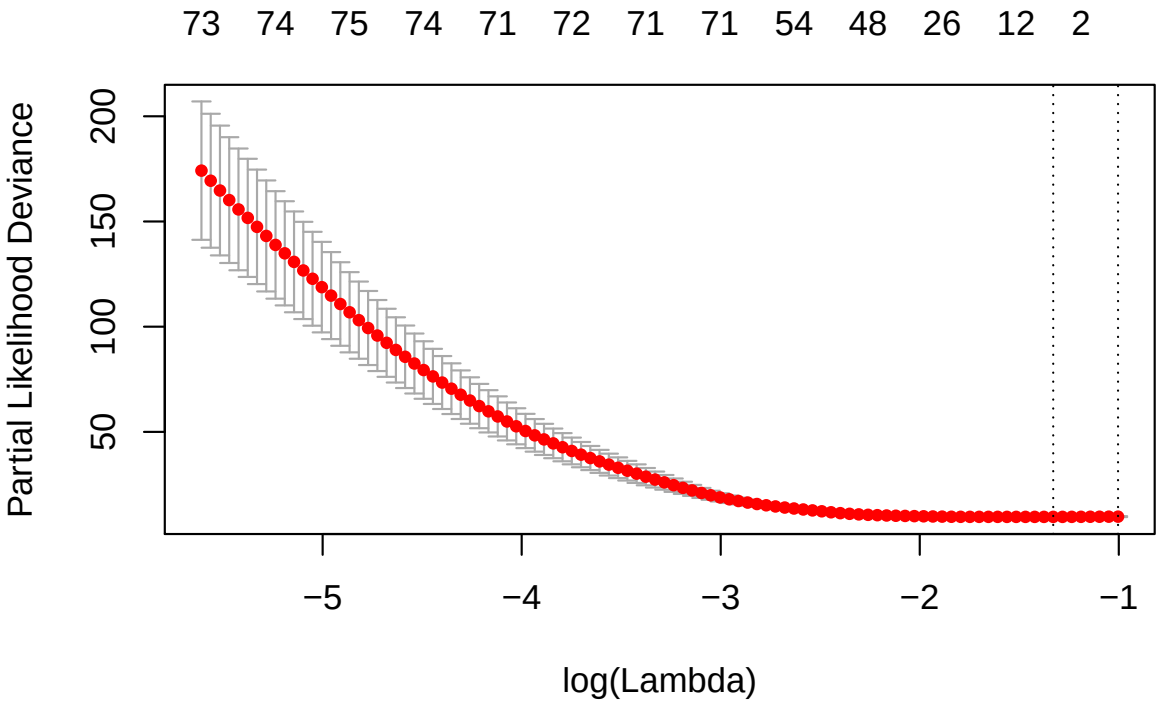

Figure 1: Cross-validated partial likelihood function, including upper and lower standard deviations, as a function of $\log \lambda$ for the AML data set.

2 Lasso penalised Cox PH regression model

We tune the lasso penalty parameter by 10-fold cross-validation using the cross-validated partial log-likelihood function as the loss function. The resulting penalty parameter value leads to a final lasso model with 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.

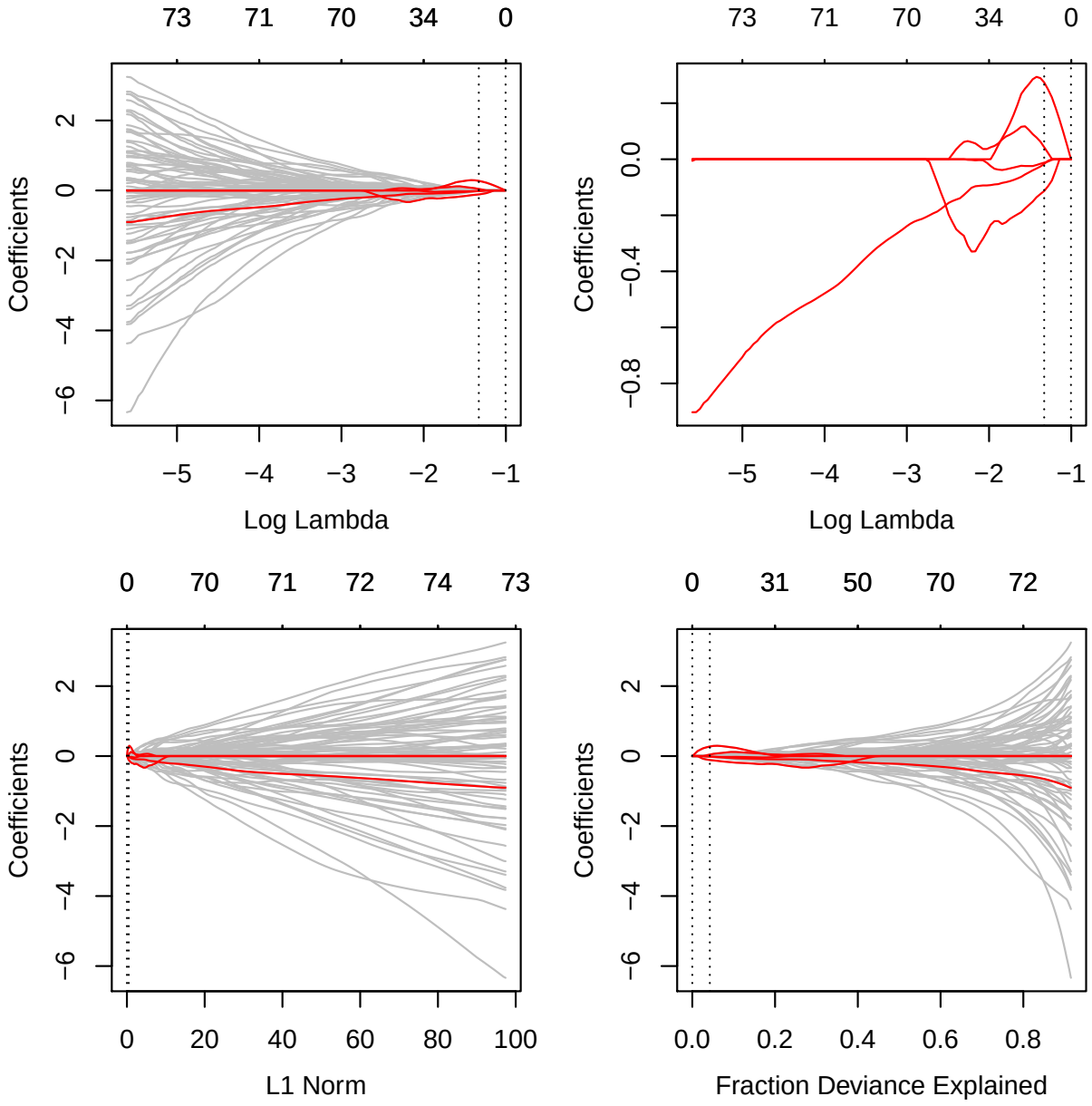

Figure 2: Coefficient paths for lasso penalised Cox PH regression model applied to the AML data set.

At this point we would like to assess the prediction performance of the lasso model. We can do this with bootstrapped prediction error curves and corresponding integrated Brier score values (see Thomas' functions adapted for peperr/pec).

Once we have seen that this model is not very satisfactory, we can attempt to improve the model in two ways. First, we can fit an elastic net model rather than lasso (and use Natalia's search algorithm for that). And second, we can assess the stability of the lasso (and elastic net) models by stability selection and identify the most stable features (using Martin's stabilityselection.R script).

3 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.

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 20 bootstrap samples for illustration.

```
> peperr_obj <- peperr(response=Surv(eset$os, eset$os_status), x=t(exprs(eset)),
+                      fit.fun=fit.glmnet,args.fit=list(standardize=F, family="cox"),
+                      complexity=complexity.glmnet, args.complexity=list(standardize=F, famil
+                      trace=F, RNG="fixed",seed=0815,
+                      indices=resample.indices(n=dim(eset)[2], sample.n = 20, method = "sub63
```

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.

The `peperr` package is designed for high-dimensional covariates data and allows for various types of parallel computations. Here, we re-run the calculations on 3 CPUs in parallel using a socket cluster on a Windows OS.

```
> peperr_obj_parallel <- peperr(response=Surv(eset$os, eset$os_status), x=t(exprs(eset)),
+                               fit.fun=fit.glmnet,args.fit=list(standardize=F, family="cox"),
+                               complexity=complexity.glmnet, args.complexity=list(standardize
+                               trace=F, RNG="fixed",seed=0815, cpus=3, parallel=T, clustertyp
+                               load.list=list(functions=c("basesurv")),
+                               indices=resample.indices(n=dim(eset)[2], sample.n = 20, method
```

Additional arguments can be passed directly to the `glmnet` call by specifing additional arguments for the fitting 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.

```
> peperr_obj_parallel_mandatory  <- peperr(response=Surv(eset$os, eset$os_status), x=data.fram
+                                          fit.fun=fit.glmnet,args.fit=list(standardize=F, fa
+                                                                           penalty.factor=re
+                                          complexity=complexity.glmnet, args.complexity=list
+
+                                          trace=F, RNG="fixed",seed=0815, cpus=3, parallel=T
+                                          load.list=list(functions=c("basesurv")),
+                                          indices=resample.indices(n=dim(eset)[2], sample.n
```

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.miscl`) 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.

For illustration purpose only we use survival status as binary response variable and build the L_1-penalized logistic regression model with `glmnet`. All three prediction accuracy measures are

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

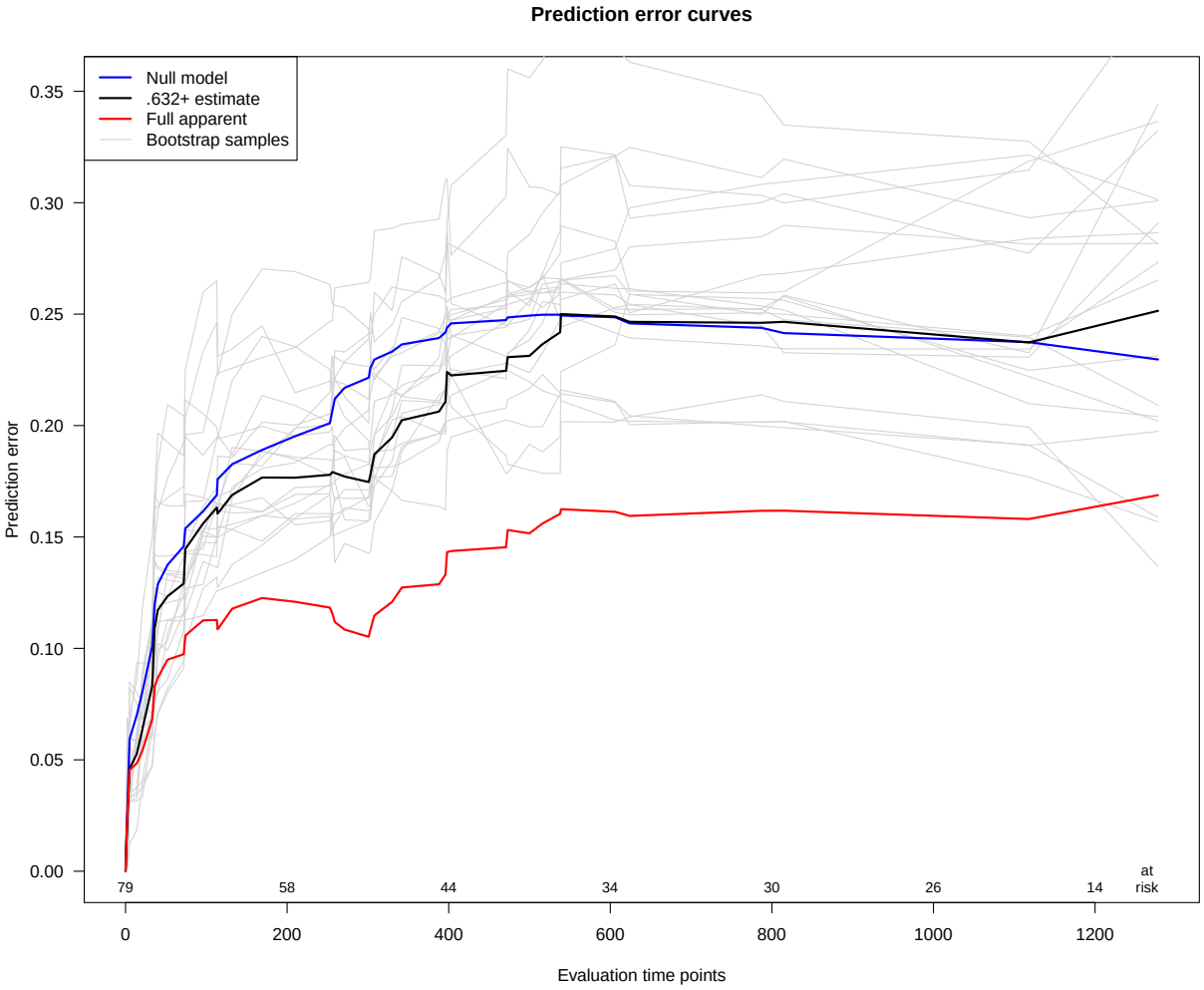

Figure 3: Prediction error curves

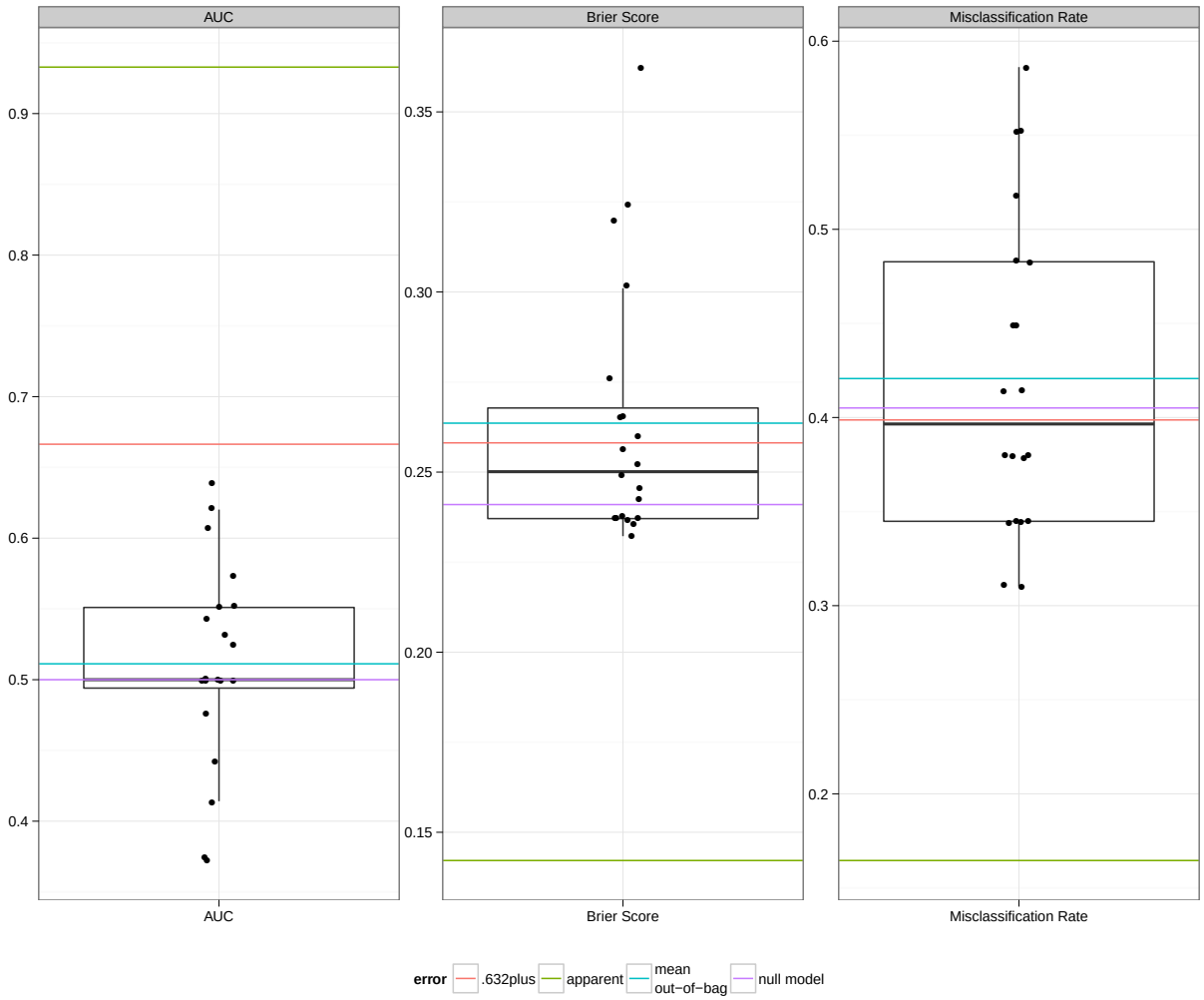

Figure 4: Different bootstrap performance measures for binary classification

calculated. Again, computations are run in parallel. Results of 20 bootstrap runs are displayed in plain boxplots (figure 4).

4 Elastic net penalised Cox PH regression model

5 Stability selection

Stable features (with $\hat{\Pi} > 0.5$ at λ =0.115) are:

```
206932_at
     2823
```

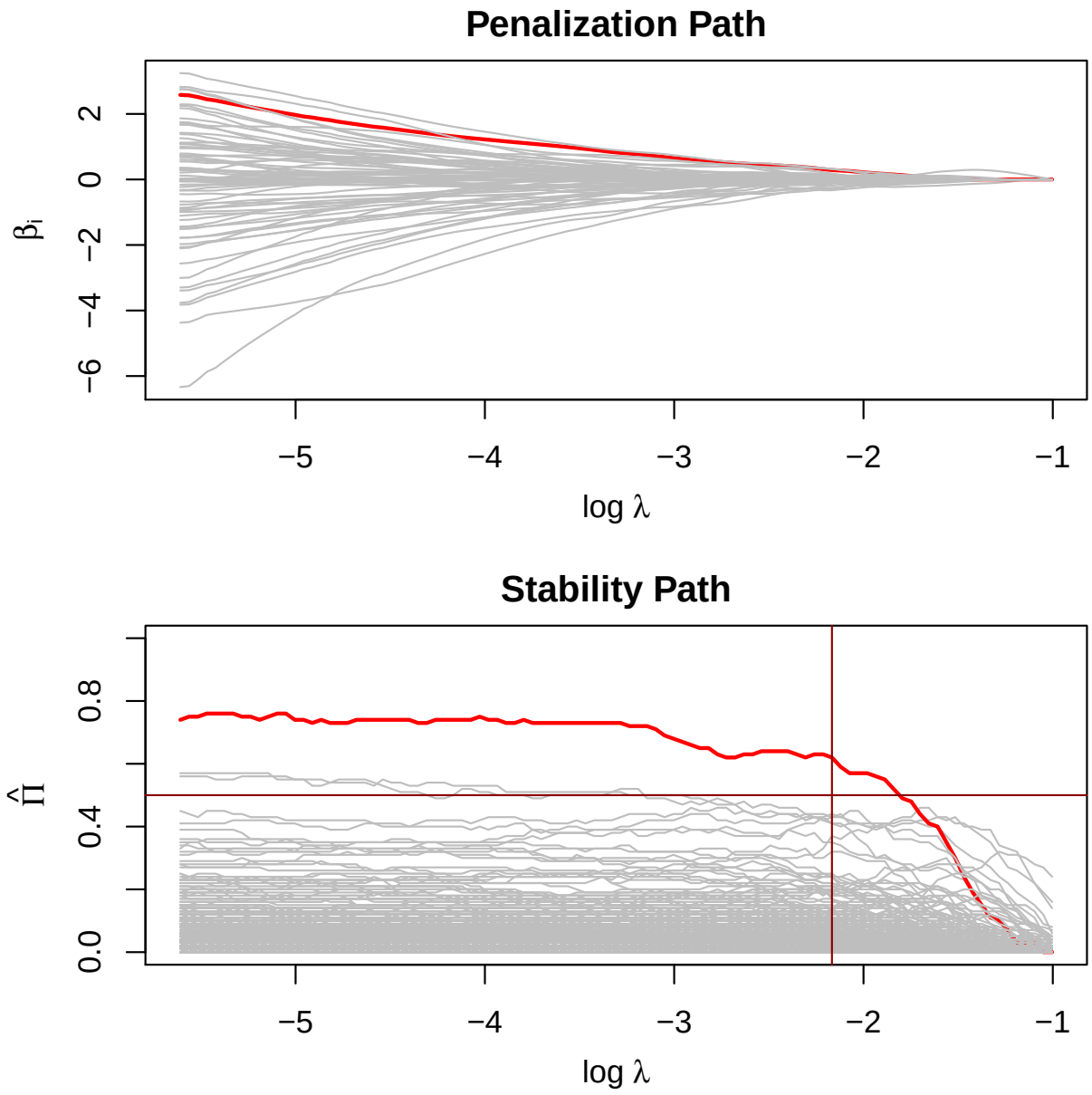

Figure 5: Coefficient and stability paths for lasso penalised Cox PH regression model applied to the AML data set.

6 Summary

7 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.15.0 (2012-03-30), `x86_64-pc-mingw32`
- Locale: `LC_COLLATE=German_Switzerland.1252`, `LC_CTYPE=German_Switzerland.1252`, `LC_MONETARY=German_Switzerland.1252`, `LC_NUMERIC=C`, `LC_TIME=German_Switzerland.1252`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, splines, stats, utils
- Other packages: Biobase 2.16.0, BiocGenerics 0.2.0, BiocInstaller 1.4.6, cacheSweave 0.6-1, codetools 0.2-8, filehash 2.2-1, genefilter 1.38.0, GEOquery 2.23.5, ggplot2 0.9.1, glmnet 1.7.4, lattice 0.20-6, limma 3.12.1, locfit 1.5-8, Matrix 1.0-6, peperr 1.1-6, snow 0.3-9, snowfall 1.84, stashR 0.3-5, survival 2.36-14
- Loaded via a namespace (and not attached): annotate 1.34.0, AnnotationDbi 1.18.1, colorspace 1.1-1, CoxBoost 1.3, DBI 0.2-5, dichromat 1.2-4, digest 0.5.2, grid 2.15.0, IRanges 1.14.3, labeling 0.1, MASS 7.3-18, memoise 0.1, munsell 0.3, plyr 1.7.1, proto 0.3-9.2, RColorBrewer 1.0-5, RCurl 1.91-1.1, reshape2 1.2.1, RSQLite 0.11.1, scales 0.2.1, stats4 2.15.0, stringr 0.6, tools 2.15.0, XML 3.9-4.1, xtable 1.7-0

References

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(1)

Graf E, Schmoor C, Sauerbrei W, Schumacher M (1999) Assessment and comparison of prognostic classification schemes for survival data. Statistics in Medicine 18(17-18):2529–2545

Porzelius C, Binder H, Schumacher M (2009) Parallelized prediction error estimation for evaluation of high-dimensional models. Bioinformatics 25(6):827–829