aBioMarVsuit: A Biomarker Validation Suit for Predicting Survival Using Gene Signature

by Pushpike J. Thilakarathne, Ziv Shkedy, Martin Otava, Willem Talloen, Luc Bijnens and Adetayo Kasim.

Abstract R package **aBioMarVsuit** can be used to discover predictive gene signature for predicting survival and dividing patients into low or high risk groups. Classifiers are constructed as linear combination of important genes and prognostic factors and treatment effects can be incorporated, if necessary. Several classifiers are implemented along with the validation procedures: majority votes technique and LASSO and Elastic net based classifiers and as function of scores of first PCA or PLS methods. Gene expression matrix is reduced using the dimension reduction methods PLS and PCA, when only scores of the first component are used in the classifier. Sensitivity analysis on the cutoff values used for the classifiers which are based on PCA and PLS can be carried out. Large scale cross validation can be performed in order to investigate the mostly selected genes during the evaluation process and therefore distributions for hazard ratios (HR) for the low risk group can be approximated both on test and training data. The inference is based on resampling methods, permutations in which null distribution of the estimated HR is approximated. Package depends on several other packages mainly, **superpc** and **glmnet**.

Introduction

Predicting patient survival and classifying them into low or high risk groups according to their survival time and other prognostic factors are widely used in cancer research. The well known Cox proportional hazard regression model is used with a few covariates and many observations (patients). Though the patients can be divided into subgroups based on their survival times, the resulting subgroups may not be biologically meaningful. Our intention is to incorporate genomic information when dividing the patients into subgroups. However, the major issue related to use of genomic information is the number of covariates, p, exceeding the number of observations, N. Special techniques need to be used to face the problem of how to predict the survival probabilities. In the literature, the main approach in this setting is based on the dimension reduction methods. For instance Beer et al. (2002) tries to use gene-expression profiles to predict survival of patients with lung adenocarcinoma using scores of first principal component obtained using PCA. Bair and Tibshirani (2004) introduces a semi-supervised methods to predict patient survival from gene expression data and Bair et al. (2006) propose prediction by supervised principal components and classification. Simon (2006) introduces therapeutically relevant predictive classifiers using gene expression profiling, and Chen et al. (2008) use supervised PCA with gene enrichment methods to predict patients survival. Partial least squares are another existing method in this setting. Nguyen and Rocke (2002) uses PLS to classify patients into low or high risk groups while incorporating the gene expressions data. All the mentioned methods mainly depend on the ordinary PCA, supervised PCA or PLS methods. Alternatively, shrinkage methods, LASSO and Elastic net, can be used to handle high dimensional and low sample size settings (Gui and Li, 2005). However, there are no proper validation schemes available for these exiting methodologies. On the other hand, there is still room for other potential methods that can be considered to classify patients as low or high risk based on their gene expression profiles and predict the survival time. Software is lacking to extract the top genes that are predictive and visualize the results in a user friendly manner.

Major goals of this particular package are to identify the genes that are predictive, estimate the HR ratios for low or high risk groups, evaluate the results using larger scale cross-validation methods and provide them in clear and visually satisfying way. In this R package, we introduce several procedures to properly identify and validate the predictive gene signature that can be used in predicting and classifying cancer patients. Mainly, we incorporate exiting statistical methods and wrap them together with available R packages. Apart from the methods based on supervised PCA and PLS, we include the machine learning method of majority votes classifier for which gene by gene analysis is performed. The well know shrinkage methods, LASSO and Elastic net for Cox proportional hazard models are also provided. All these methods are evaluated using ℓ-fold cross-validation and inner cross-validation methods. It is interesting to monitor and visualize the genes that tend to be selected most often during the ℓ-fold cross-validations based on shrinkage methods. Further, functions are implemented to approximate the null distribution of the HR for low risk group. Then, the we can test how

extreme is the estimated HR of the original data compared to the estimated HR on the permuted data. In the following section we elaborate more on the technical details of the methodologies used in the package.

Methodology

The package is built up around the methods that we discussed in the previous section. First we deal with the curse of dimensionality. Main issue in fitting Cox proportional model (as coxph in following text) is that we have larger number of predictors (genes) than the number of patients (N). Possible solution is applying dimension reduction methods. For instance scores of the first principal component of PCA can be used as a predictor in the coxph model instead of gene expressions or supervised PCA can be used to identify the most interesting genes that strongly correlated with survival (Bair et al., 2006). The latter approach is normally done by fitting a coxph model for each of the genes separately and ranking them based on the Cox scores. In this way, we can reduce the gene expression matrix into a smaller one, preferably of 100 to 150 genes. Note that top genes to retain can set by user. For this particular package there is an option to use either entire gene expression matrix or matrix reduced by SPCA. Nevertheless, it is recommended to work already on the reduced gene expression matrix as it is known that only few genes are responsible for predicting survival and this is usually case with microarray experiments. Another dimension reduction approach is the PLS, its implementation in combination with coxph model is proposed by Devarajan et al. (2010). In this package, for both methods are available for ℓ-fold cross-validations, sensitivity analysis on the cutoff value, approximation of null distribution of the HR and visualization.

Our methodology to classify patients into low or high risk group and estimating the hazard ratio are as follows. Let $\mathbf{T}$ be time to event of interest, such as survival time for diseased patients or disease free time. The p covariates associated with each patient let be gene expression levels of p genes. Suppose we have a sample size of N and due to censoring, for $i = 1,\ldots,N$, the i^{th} observation in the sample is denoted by $(T_i, \delta_i, X_{i1},\ldots,X_{ip})$, where δ_i is the censoring indicator and T_i is the survival time if $\delta_i = 1$ or censored time if $\delta_i = 0$, and $\mathbf{X}_i = (X_{i1},\ldots,X_{ip})'$ is vector of gene expression levels of p genes for the i^{th} subject. Let $\mathbf{h}(t)$ be the hazard function at time t and let $P = (P_1,\ldots,P_m)$ be the vector of prognostic factors. Then, we define following Cox proportional hazard model at time t

$$\begin{aligned} h(t) &= h_o(t)\exp(\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \gamma_1 P_1 + \ldots + \gamma_m P_m) \\ &= h_o(t)\exp(\mu) \end{aligned} \tag{1}$$

Where $h_o(t)$ is the baseline hazard function. However, due to the high dimensional space of the genes with expression levels (predictors), the standard maximum Cox partial likelihood methods cannot be applied directly to fit the model (1). Therefore, we replace the linear predictor μ by $\mathbf{U}(.)\beta + \sum_{m=1}^{m} P_m\gamma_m$ or $\sum_{k=1}^{K} X_k\beta_k^s + \sum_{m=1}^{m} P_m\gamma_m$, where $\mathbf{U}(.)$ is the function of gene expressions either based on reduced $\mathbf{X}_r$ or full gene expression matrix $\mathbf{X}$. For instance $\mathbf{U}(.)$ can be a linear function of either

- scores of the first PCA,
- scores of the first PLS,
- linear combination of genes with weights using LASSO or Elastic Net approaches,
- single gene expressions.

The package by default reduce the gene expression matrix by using the supervised PCA approach via **superpc**. However, it can be controlled by the user by passing the appropriate argument to the function calls.

Once the coxph model is fitted, we can estimate the risk of the patients as a function of $\mathbf{U}(.)$. We define linear risk function $\mathbf{R}(x)$ in different ways depending on the method to replace the linear predictor μ, provided that no prognostic factors are available. If there are prognostic factors, they are added automatically to the linear predictor such that the risk scores can be adjusted for them.

$$\mathbf{R}(x) = \begin{cases} U(X_r)\beta_{PCA1} & \text{if first PCA is used} \\ U(X_r)\beta_{PLS1} & \text{if first PLS is used} \\ U(X_k)\beta_k & \text{if gene k is used} \\ \sum_{k=1}^{K} X_k\beta_k^s & \text{if shrinkage methods are used and } K \ll p \end{cases}$$

If $\mathbf{R}_i(x) > \text{median}[\mathbf{R}_i(x)]$ then patient i is labeled as having high risk, otherwise having low risk. Let $\mathbf{Z}_i$ be the indicator variable which indicates i^{th} subject is having low or high risk such that,

$$\mathbf{Z}_i = \begin{cases} 1 & \text{low risk} \\ 0 & \text{high risk} \end{cases}$$

The estimated risk score $\widehat{\mathbf{R}(x)}$ can be used to label patients into low or high risk groups and hazard ratio for low or high risk group can be estimated by refitting the coxph model with the indicator variable $\mathbf{Z}_i$ as a covariate in the model. Usually median risk is used as a cutoff value, but other options can be considered.

If $\mathbf{R}(x)$ is linear function of only single gene specific expression level then a single patient can be classified into low or high risk group as many times as genes used. For instance if we use top 100 genes then a particular patient can be labeled 100 times. Therefore, final group for that particular patient is chosen based on the majority votes. That is, final labels for patients can be done as follows (for K genes taken):

$$\mathbf{Z}_i = \begin{cases} 1 & \text{low risk} \quad \text{if } \frac{1}{K}\sum_{j=1}^{K} Z_{ij} \geq 0.5 \\ 0 & \text{high risk} \quad \text{if } \frac{1}{K}\sum_{j=1}^{K} Z_{ij} < 0.5 \end{cases}$$

.

It is difficult to identify or rank predictive genes based on the risk function based on first PLS or PCA scores. However, supervised PCA can be used to rank genes and thereby reduced the gene expression matrix that can be used in the downstream analysis. Prediction methods based on gene by gene analysis and shrinkage methods allow genes to be ranked by using gene specific hazard ratio or the selection frequency of genes during the ℓ-fold cross-validations under shrinkage methods. To that end we have implemented functions to perform gene by gene analysis and shrinkage methods.

Other possible analysis included in the package is method when we sequentially increase the number of top genes, found by gene by gene analysis, to form the reduce gene expression matrix, derive the first PCA or PLS on that matrix and label the patients based on the risk score. Hazard ratio can be estimated afterwards for each top $K = K_1, \ldots, K_n$ genes. Hence, we can investigate variability of the HR over different top K genes. The whole analysis can be afterwards cross-validated. The cross-validation schemes used in this package will be discussed in following section. Apart from the univariate approaches, package has ability to build predictive models using LASSO (Hastie et al., 2001) and Elastic net methods (Zhou and Hastie, 2004) for which either reduced or full gene expression matrix can be used (Tibshirani, 1994). To that end, we built up several wrapper functions around the package **glmnet** functions.

ℓ-fold cross-validations

Cross-validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. We will focus on ℓ-fold cross-validations in which dataset is divided into ℓ sub datasets. For instance, if ℓ=3 then dataset is divided into three groups and $\frac{2}{3}$ of the data (training set) is used to fit the model and label the patients and $\frac{1}{3}$ of the data (test set) is used to validate the results. Package includes the functions that enable to perform cross-validations many many times and derive distribution of the HR for low risk group on the training and test data. The ℓ-fold cross-validation is implemented for all methods that we discussed in the previous section. In particular, LASSO and Elastic net based predictive models are cross-validated in following way. First, data are divided into in bag ($N\frac{2}{3}$) and out of bag ($N\frac{1}{3}$) data. In bag data are further partitioned into training and test data in which inner cross-validations are carried out many many times. Selected genes during the inner cross-validation process are recorded and HR ratios are computed on the test data within each inner iteration. The mostly selected genes during the inner cross-validations are used to built the model on the out of bag data for which hazard ratio is also computed. The entire process can be repeated many many times on the top K genes. In this way we can investigate the distribution of the HR on the out of bag and in bag data simultaneously. It is also possible to identify the genes that tend to be selected during this cross-validations process. That information can be used to rank genes to be used in the downstream analysis. Moreover, genomic classifier can be developed based only on these selected genes. Let $\mathbf{R}(x)$ be the classifier or risk score which is a linear function of genes.

$$\mathbf{R}(x) = \sum_{k=1}^{K} w_k X_k$$

for which $K \ll p$ and weights $w_k = \frac{1}{B}\sum_{b=1}^{B} \beta_{bk}^s$ where β_{bk}^s is the shrinkage coefficients of the k^{th} gene at the b^{th} cross-validation. Median of the risk score can be used to classify patients into low or high risk group.

Classifier is the risk score which is a linear function of genes which are selected mostly. Weights for each gene can be estimated as a mean of the shrinkage coefficients that are estimated during the inner cross-validation process. Alternatively, we can use majority vote based method to label patients. Let $\mathbf{R}_i(x)_b$ be risk score or classifier at the b^{th} cross-validation for patient i defined as

$$\mathbf{R}(x)_b = \sum_{k=1}^{K} \beta_{bk}^s X_k,$$

then patient i can be labeled into groups as

$$\mathbf{Z}_{ib} = \begin{cases} \text{if } R_i(x)_b > \text{median}(R(x)_b) & \text{high risk} \\ \text{if } R_i(x)_b \leq \text{median}(R(x)_b) & \text{low risk} \end{cases}$$

and Z_{ib} is the indicator variable showing that the i^{th} subject is classified either low or high risk at b^{th} cross-validation.

It means, at each iteration we have a classifier with certain weights (shrinkage coefficients) and labels for patients having low or high risk. If we repeat this at each iteration we can finally find the group for each patient based on the majority votes. All these methods are implemented as functions in this particular package and examples are given.

Moreover, performs of the leave-one-out cross-validation (LOOCV) can be investigated. LOOCV is a special case of ℓ-fold cross-validation where ℓ equals N (Stone, 1974). In other words in each iteration nearly all the data except for a single observation are used for training and the risk score can be estimated for the single subject. Cutoff value is the median risk score obtained under training set. This method is widely used when N is very small, especially in bioinformatics where only dozens of data samples are available.

Null distribution of the HR

Apart from the cross-validation methods described in the previous section, the permutation analysis to assess the results of the genomic classifiers can also be carried out. In particular, goal is to approximate the null distribution of the HR based on different classifiers. To that end, we break the relationships between survival time and the gene expression by randomly shuffling the observation of the response vector $\mathbf{T}$ while keeping the gene expression matrix $\mathbf{X}$ as it is. Classifier is then built using the methods described in the previous section and HR is estimated afterwards. This procedure is repeated many many times. Under the null distribution, we expect estimated HR to be around 1 which implies that there is no unique gene signature that can be used to identify the low and high risk patients if the relationship between gene expression and the survival is weak. Let $\widehat{HR_{low}}$ be the estimated HR for the low risk group using the original dataset and let $\widehat{HR_{low}^b}$ be the HR estimated at b^{th} permuted dataset. Hence, the empirical p-value is defined as

$$\tilde{p} = \frac{1}{B} \sum_{b=1}^{B} \mathbf{I}\left(\widehat{HR_{low}} < \widehat{HR_{low}^b}\right) \tag{2}$$

where B is the number of permutations and $\mathbf{I}$ is an indicator function. This p-value gives an idea how extreme is the estimated HR compared to the HR under the permuted data.

Introduction to R Package aBioMarVsuit

The main functions of the **aBioMarVsuit** are `InnerCrossValELNet()`, `CvForMajorityVotes()`, `CVSeqIncreaseGenes()`, `CVforDimPcaPls()` and `NullDistHR()`. The function `InnerCrossValELNet()` can be used to perform cross-validations on shrinkage methods during which we can monitor genes that tend to be selected. In this particular function is performed inner and outer cross-validations. Classifier is built as linear function of predictive genes and corresponding weights are estimated as average of the shrinkage coefficients that are estimated during the inner cross-validations. Patients are labeled according to either the risk score based on weighted linear combination of the predictive genes or majority votes based methods (see Section 1). The function `CvForMajorityVotes()` can be used to perform the cross-validation on majority votes based classifier which internally perform the gene by gene based labeling. The function `CVSeqIncreaseGenes()` further processes the cross-validated gene by gene analysis results. Function sequentially considers top $K = K_1, \ldots, K_n$ number of genes based on the estimated HR on the test data of the cross-validated gene by gene analysis (see more detail about the functionality of this function in Section 1). The goal of this particular function to investigate the distribution of the HR over different top K genes. The function `CVforDimPcaPls()` performs cross-validations for the analysis performs by `SurFitPlsClasif()` and `SurFitPcaClasif()` functions where the risk score is built up as a linear function of scores of first PCA or PLS component.

All these main functions are internally using several other functions, either supporting functions or functions where the analysis can be carried out without cross-validations. Outputs from all the main and most

Category	Functions	description
Basic	`GeneSpecificCoxPh`	Gene by gene Cox proportional hazard model
	`SurFitPcaClasif`	Classifier based on first PCA
	`SurFitPlsClasif`	Classifier based on first PLS
	`MajorityVotes`	Majority votes classification
	`LassoElasticNetCoxPh`	Wrapper function for **glmnet**
advance	`CVLassoElasticNetCoxPh`	Cross-validations for LASSO and Elastic net based predictive models
	`CVSeqIncreaseGenes`	Cross-validation for Top $K_1,\ldots,K_n$ genes
	`CVforDimPcaPls`	Cross-validations for PCA and PLS based methods
	`CvForMajorityVotes`	Cross-validation for majority votes
	`GeneOccurence`	Mostly selected genes by cross-validated gene by gene analysis
	`GridAnalysis`	Sensitivity of the cutoff values used in classification
	`InnerCrossValELNet`	Inner and outer cross-validations for shrinkage methods
	`NullDistHR`	Null distribution of the estimated HR
	`SeqIncreaseGenes`	Sequentially increase the number of top K genes

Table 1: Basic and advance R functions in **aBioMarVsuit**

of supporting functions is returned as `S4 class` objects which allow user to use standard generic functions such as `show`, `summary`, and `plot`. The summary of the functions and their descriptions are presented in Table 1. The **aBioMarVsuit** can be obtained at http://cran.r-project.org/web/packages/aBioMarVsuit/index.html. The package depends on **glmnet**, **survival**, **PLS**, and **superpc** packages.

Illustrations using a real-life dataset

To demonstrate the usage of the functions in this package, we used publicly available dataset of DLBCL by Rosenwald et al. (2002). This dataset includes a total of 240 patients with DLBCL, and their gene expression data over 7399 genes. However, patients having missing survival time are removed and genes having missing expression levels for some patients are also removed prior to use in the analysis. Therefore, gene expression matrix without any missing values does have 179 patients (samples) and 3583 genes.

```
library(aBioMarVsuit)
data(exprLym)
GexprMatrix <- exprs(exprLym)
SurvData <- pData(exprLym)
PatietId <- rownames(SurvData[!is.na(SurvData[, c("IPI")]), ])
Gdata <- GexprMatrix[, PatietId]
```

Following sections show few examples of applying the basic and advance functions to analyze this data.

Basic functions and their usage

In this section we introduce a set of functions that can be used to reduce the gene expression matrix, classify and estimate HR for low risk group without employing any cross-validations.

PLS based method

First, the function `SurFitPlsClasif()` reduces larger gene expression matrix to smaller one using supervised PCA approach. Note that this is an parameter in function and entire gene expression matrix can be used instead. This function performs the PLS on reduced gene expression matrix and fit Cox proportional hazard model with first PLS scores as a covariate in the model. Then, classifier is built based on the first PLS scores multiplied by its estimated regression coefficient. Patients are classified using median of the risk scores.

It can also be used to perform the grid analysis where the grid will be several cutoff values and default is median cutoff. This function can handle single and multiple genes. Other prognostic factors can be incorporated to the model. Figure 1 shows the results based on this function.

```
results1 <- SurFitPlsClasif(SurvTime, Gdata, Censor, ReduceDim=TRUE, NuFeToBeSel=100,
    ProgFact=NULL, Plots = TRUE, MedianCut = NULL )
```

```
summary(results1$SurFit)[[8]][-2]
```

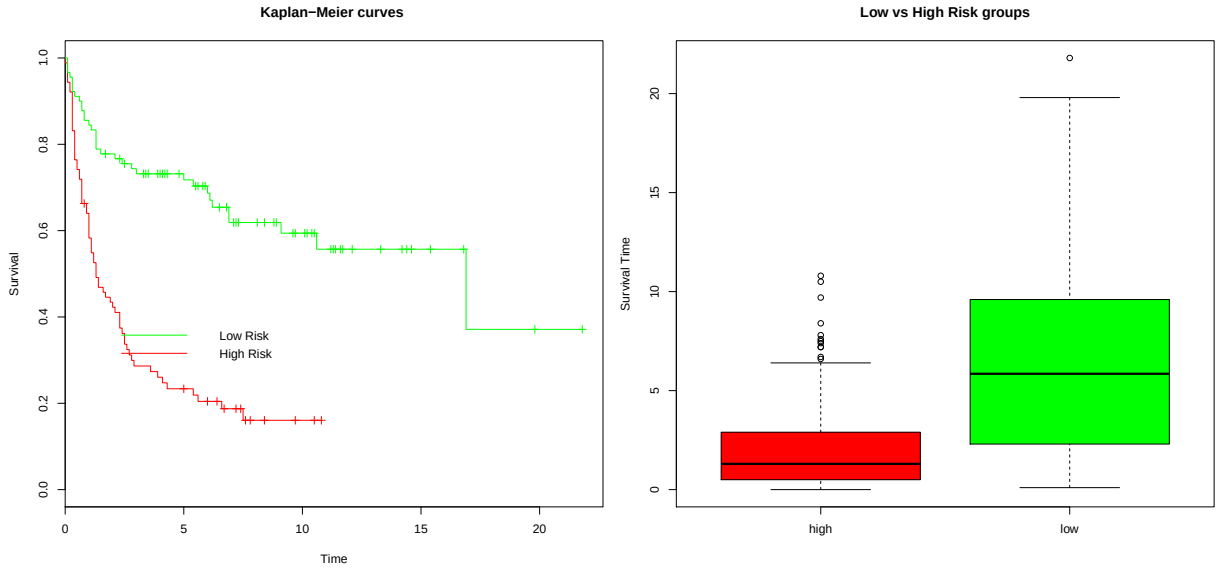

Figure 1: Distribution of the Survival time and the survival probabilities by Kaplan-Meier curves for low and high risk groups. Note that, patients are labeled using the risk score estimated based on the PLS based method.

Gene by Gene CoxPh modelling

Another option in the package is the `GeneSpecificCoxPh()` by which gene by gene analysis can be performed. In particular function fits gene by gene Cox proportional hazard model and perform the classification based on median risk score which has been estimated using a single gene expression level.

```
Ana1 <- GeneSpecificCoxPh(SurvTime, Gdata, Censor, ReduceDim=TRUE, NuFeToBeSel=50,
                          ProgFact=NULL, MedianCut = NULL)
show(Ana1)

Gene by Gene CoxPh Model
Number of Genes used:  50

summary(Ana1)

Summary of Gene by Gene CoxPh Models
Number of Genes used:  50
Top 15 Genes out of  50
Estimated HR for low risk group
          GeneNames        HR       LCI       UCI    FDRLCI    FDRUCI
1       ITGA6(3655) 0.3953318 0.2641103 0.5917499 0.2917337 0.6536413
2  SERPINA9(327657) 0.3970800 0.2651714 0.5946061 0.2927673 0.6564856
3      ASB13(79754) 0.4285292 0.2869030 0.6400674 0.3141830 0.7009279
4        IRF4(3662) 0.4466176 0.2987995 0.6675624 0.3257638 0.7278047
5      GFOD1(54438) 0.4535518 0.3041587 0.6763220 0.3310595 0.7361382
6         CR2(1380) 0.4630911 0.3119120 0.6875444 0.3387419 0.7466854
7       PRKD1(5587) 0.4807427 0.3234805 0.7144589 0.3498996 0.7728100
8         TOX(9760) 0.4904690 0.3296994 0.7296339 0.3558616 0.7875317
9       TCEB3(6924) 0.4975821 0.3347007 0.7397295 0.3607027 0.7971970
10       LMO2(4005) 0.5029901 0.3391021 0.7460850 0.3650228 0.8031152
11        BCL6(604) 0.5084289 0.3427233 0.7542525 0.3684958 0.8109716
12       MRC1(4360) 0.5114390 0.3442668 0.7597881 0.3699212 0.8164067
```

```
13    COL16A1(1307) 0.5141666 0.3476782 0.7603792 0.3733739 0.8165762
14    GCET2(257144) 0.5270380 0.3560496 0.7801414 0.3813520 0.8355815
15        BMP6(654) 0.5303285 0.3568641 0.7881103 0.3819695 0.8435538
```

Summary of this analysis can be used to select the top genes based on the HR.

```
plot(Ana1)
```

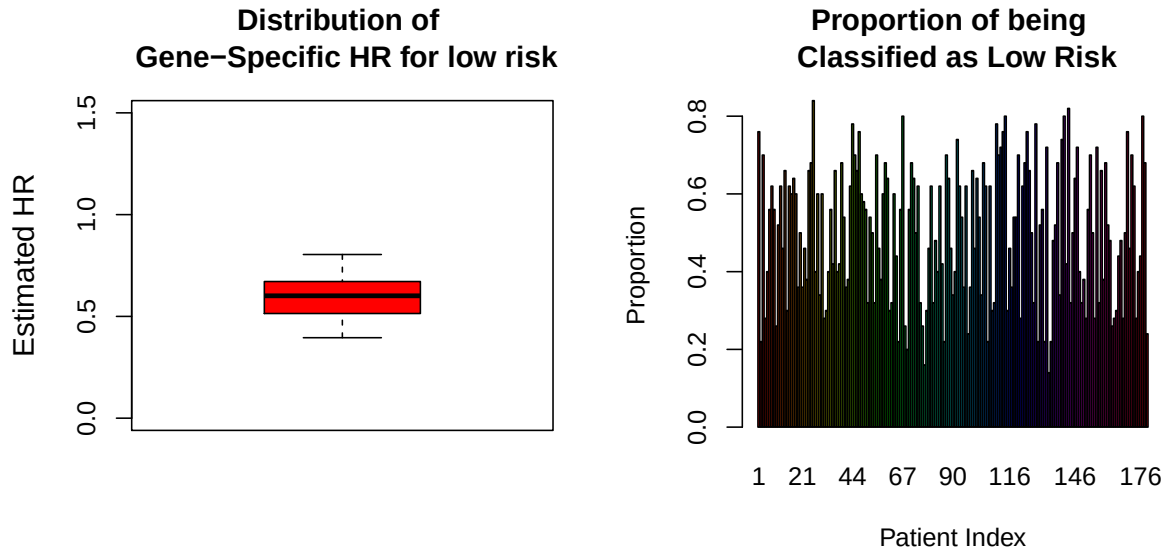

Figure 2: Left panel shows distribution of the estimated HR for low risk group. Right panel shows proportions being classified as low risk for each patient.

Majority Votes

The function `MajorityVotes()` performs the classification based on majority votes and estimate the HR for low risk group. Note that input for the this function is the output from `GeneSpecificCoxPh()` function. That is function `MajorityVotes()` internally calls function `GeneSpecificCoxPh()`.

```
MVres <- MajorityVotes(Ana1, ProgFact = NULL, SurvTime, Censor, J = 1)
```

Lasso

The function `LassoElasticNetCoxPh()` is a wrapper function for **glmnet** and fit model with all prognostic factors and genes. LASSO, Elastic net and Ridge regressions can be fitted. Optimum lambda will be used to select the non-zero shrinkage coefficients. The function can accommodate prognostic factors and they will be forced to be in the model if suitable. Then, estimated HR for low risk group, class labels for each patient and other objects are returned.

```
NuFeToBeSel = 50
DataForReduction <- list(x = Gdata, y = SurvTime, censoring.status = Censor,
  featurenames = rownames(Gdata))
TentativeList <- names(sort(abs(superpc.train(DataForReduction,
        type = "survival")$feature.scores), decreasing = TRUE))[1:NuFeToBeSel]
ReduGdata <- Gdata[TentativeList, ]
```

Application of LASSO with 50 genes

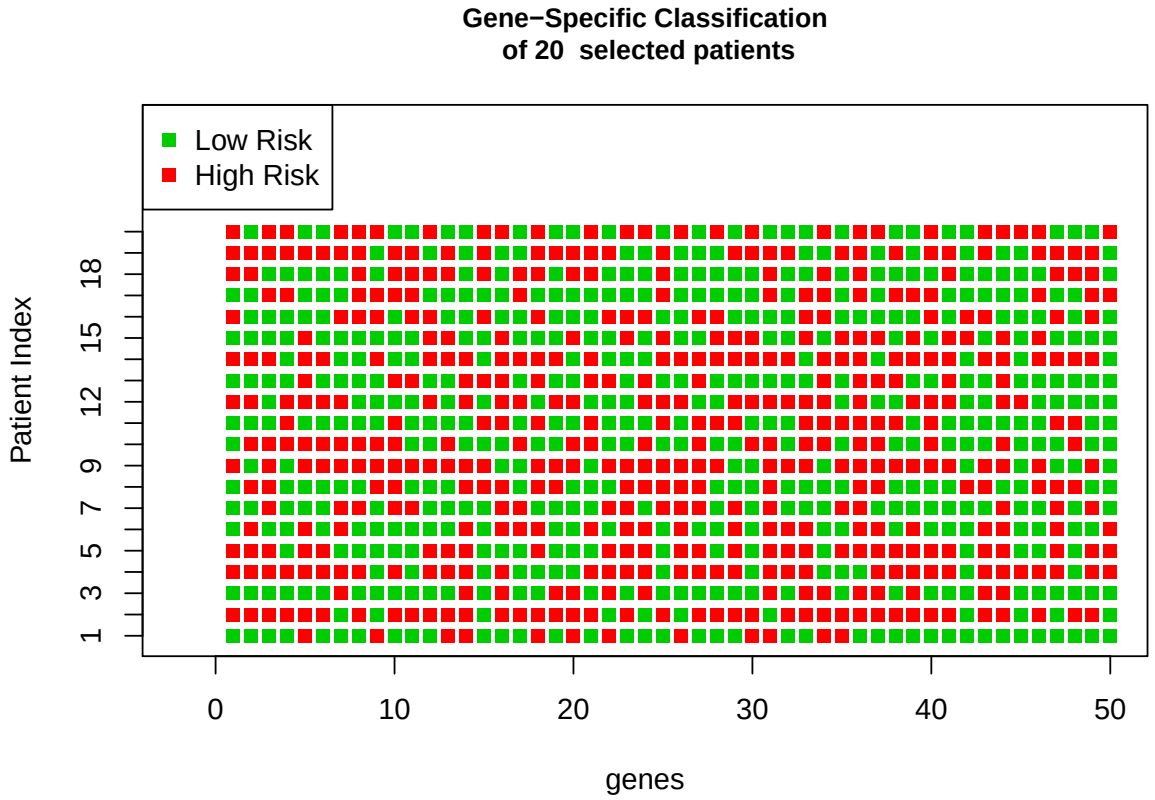

Figure 3: Majority votes classification.

```
set.seed(145)
L1 <- LassoElasticNetCoxPh(SurvTime, Censor, ReduGdata[1:50, ], ProgFact = NULL,
        Plots = TRUE, MedianCut = NULL , GeneList = NULL,
        StZ = TRUE,  alpha = 1)
summary(L1$Results$SurFit)

Call:
coxph(formula = Surv(SurvTime, Censor == 1) ~ gid, data = c2data)

  n= 179, number of events= 104

          coef exp(coef) se(coef)      z Pr(>|z|)
gidlow -2.1330    0.1185   0.2443 -8.731   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       exp(coef) exp(-coef) lower .95 upper .95
gidlow    0.1185       8.44    0.0734    0.1912

Concordance= 0.734  (se = 0.026 )
Rsquare= 0.409   (max possible= 0.996 )
Likelihood ratio test= 94.24  on 1 df,   p=0
Wald test            = 76.23  on 1 df,   p=0
Score (logrank) test = 100.6  on 1 df,   p=0
```

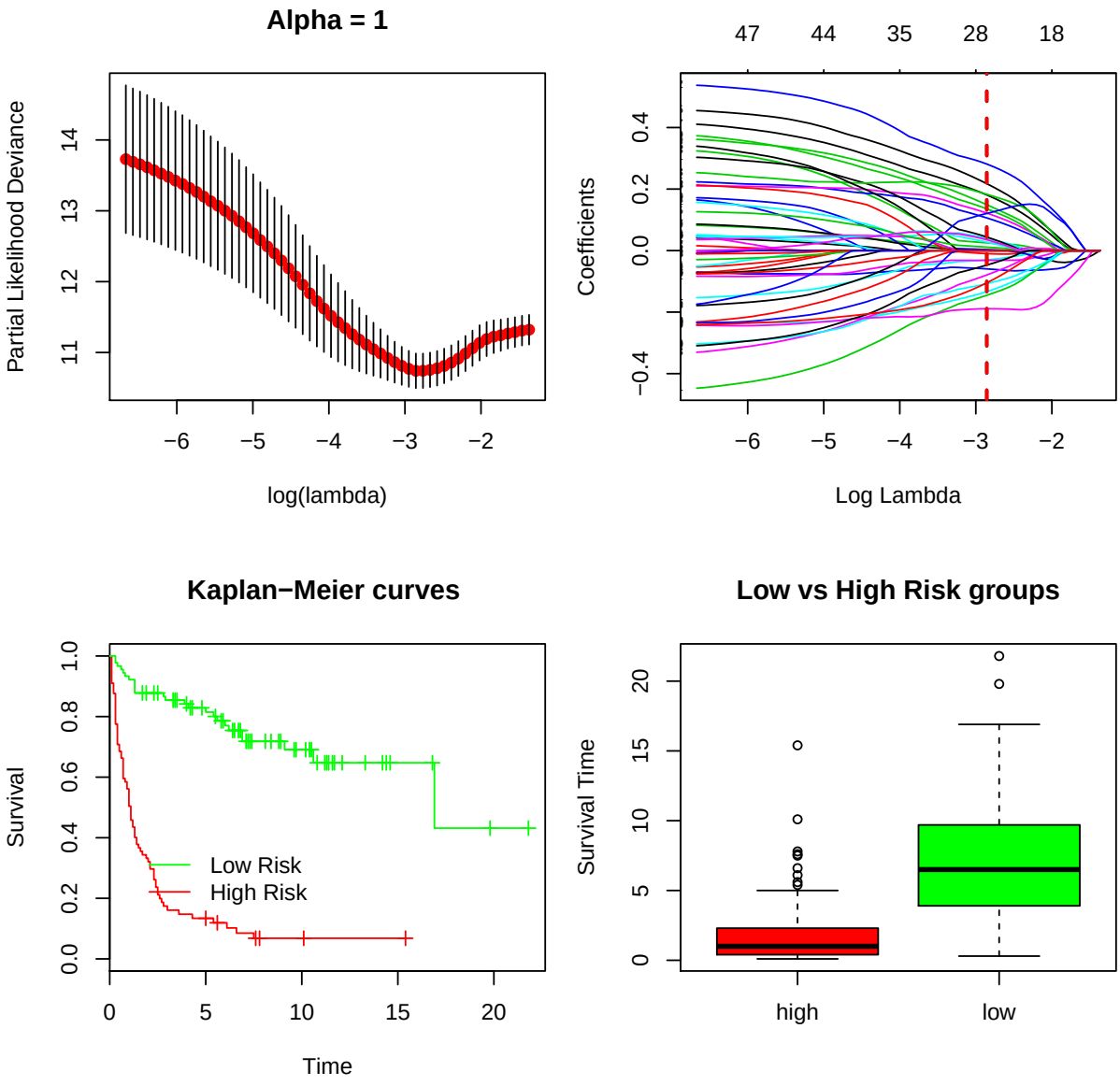

Figure 4: LASSO.

Advance functions and their usage

In this section we focus on more advanced functions that can be used for cross-validations, feature selection, and sensitivity analysis of the cutoff values for classifications and top K gene analysis.

Sensitivity Analysis of the Cutoff value

Usually, the median risk is used for as the cutoff value, however it is interesting to investigate the what will be the effect on the Hazard ratio if we use different cutoff values. The function `GridAnalysis()` performs the sensitivity of the cutoff values used in classification. The percentiles of risk score obtained under PCA or PLS methods are used as fine grid points as cutoff values.

```
GridAnalysis(SurvTime, Gdata, Censor, ReduceDim = TRUE, NuFeToBeSel = 150,
 ProgFact = NULL, Plots = TRUE, DimMethod = "PLS")
```

```
           CutOff EstimatedHR    LowerCI   UpperCI
 [1,] -0.86203595   0.1039109 0.04533646 0.2381631
 [2,] -0.70226637   0.1234498 0.06187801 0.2462888
 [3,] -0.57856137   0.1538193 0.08377212 0.2824375
 [4,] -0.53628533   0.1881593 0.10824069 0.3270853
 [5,] -0.47359621   0.2150545 0.12985297 0.3561602
 [6,] -0.39779517   0.2191468 0.13464915 0.3566700
 [7,] -0.33645313   0.2099129 0.12969555 0.3397451
 [8,] -0.20566820   0.2493762 0.15993746 0.3888301
 [9,] -0.10995699   0.2636277 0.17074378 0.4070401
[10,]  0.06500409   0.2531341 0.16649312 0.3848619
[11,]  0.27311652   0.2502934 0.16654117 0.3761640
[12,]  0.44064450   0.2485290 0.16590421 0.3723031
[13,]  0.55935161   0.2394753 0.16046993 0.3573781
[14,]  0.63258032   0.2257759 0.15084145 0.3379359
[15,]  0.71469784   0.1974028 0.13061571 0.2983399
[16,]  0.76867571   0.1860115 0.12167396 0.2843687
[17,]  0.88033374   0.1981357 0.12864528 0.3051627
```

Other dimension reduction methods can also be considered. Figure 1 shows the estimated HR along with the 95% CI at different cutoff values.

Sequentially increases the number of top *K* genes

The one of the major goal of this package is to select the top *K* genes as biomarkers to to be able to predict the survival and group patients into low or high risk groups. However, there is no rule of thumb that how many number features, genes, should retain for the gene signature. And therefore, we have built up a function that perform the analysis using different number of top *K* genes which can be used to examine the effect of number of top genes on the estimated hazard ratio. The function `SeqIncreaseGenes()` sequentially increases the number of top *K* genes to be used in the PCA or PLS methods in order to obtain the risk score.

```
SeqIncreaseGenes(TopK = 15, SurvTime = SurvTime, Gdata = Gdata, Censor = Censor, ReduceDim = TRUE,
 NuFeToBeSel = 150, ProgFact = NULL, Plots = TRUE, DimMethod = "PLS")
```

```
      Topk EstimatedHR   LowerCI   UpperCI
 [1,]    1   0.3953318 0.2641103 0.5917499
 [2,]    2   0.3813561 0.2544845 0.5714788
 [3,]    3   0.3931536 0.2620809 0.5897789
 [4,]    4   0.3903266 0.2600538 0.5858590
 [5,]    5   0.3821902 0.2547258 0.5734377
 [6,]    6   0.3551867 0.2360046 0.5345559
 [7,]    7   0.3380196 0.2239768 0.5101297
 [8,]    8   0.3710470 0.2470192 0.5573489
```

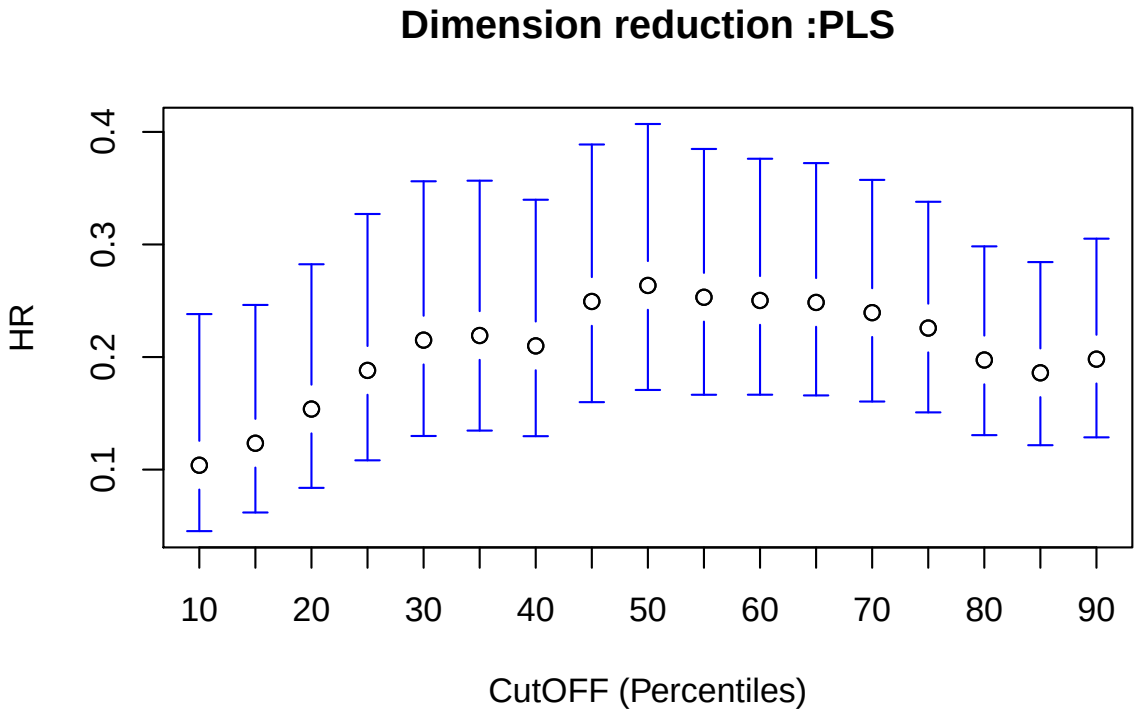

Figure 5: Sensitivity of the cutoff values used.

```
 [9,]    9   0.3568943 0.2370431 0.5373435
[10,]   10   0.3498675 0.2320054 0.5276051
[11,]   11   0.3332633 0.2203470 0.5040434
[12,]   12   0.3232763 0.2136125 0.4892390
[13,]   13   0.3236737 0.2138699 0.4898524
[14,]   14   0.3437051 0.2279618 0.5182150
[15,]   15   0.3295460 0.2183039 0.4974742
```

Cross validations for PCA and PLS methods

The function `CVforDimPcaPls()` can be used to achieve cross-validations on the analysis performed by `SurFitPlsClasif()` and `SurFitPcaClasif()` functions where the dimension reduction methods are PCA and PLS, respectively. Br defaults 3-fold cross validations are performed and others can be considered as well.

```
R1 <- CVforDimPcaPls(fold = 3, SurvTime, Gdata, Censor, ReduceDim = TRUE,
                        NuFeToBeSel = 150, ProgFact = ProgFact,
                        Plots = FALSE,  n = 50 ,  DR = "PLS", mtitle = "")
show(R1)

plot(R1, ylim=c(0, 5))
```

Cross validations for majority votes

The function `CvForMajorityVotes()` performs cross-validations for majority votes based classification. The function internally calls `MajorityVotes` and `GeneSpecificCoxPh`.

```
cvmvres <- CvForMajorityVotes(SurvTime, Censor, ProgFact = NULL, Gdata, ReduceDim = TRUE,
  NuFeToBeSel = 50, fold = 3, nCV = 50)
```

Figure 6: Estimated HR with different top *K* genes.

```
show(cvmvres)

plot(cvmvres)
```

Cross validations for gene by gene analysis

The function `CVGeneSpecificCoxPh()` performs the cross-validation for gene by gene analysis. First, data will be divided into training and test. Gene-specific model is fitted on training data and classifier is built. Then, classifier is evaluated on test data for that particular gene. Process is repeated for all genes and all these steps can be repeated many times.

```
CVR1 <- CVGeneSpecificCoxPh(fold = 3, SurvTime, Gdata, Censor,
                        ReduceDim = TRUE,  NuFeToBeSel=50,
                        ProgFact = NULL, MedianCut = NULL,
                        n.cv = 50)
show(CVR1)

Cross-validated Gene by Gene Analysis
Number of Genes used:  150
Number of cross valdiations used:  50

summary(CVR1, Which = 20)

Summary of Cross-validated Gene by Gene Analysis
Estimated Median of the cross-validated HR for Feature:  20
Estimated HR for Train Data
0.8021(0.5716-1.7034)
Estimated HR for Test Data
0.7872(0.4452-2.1934)
```

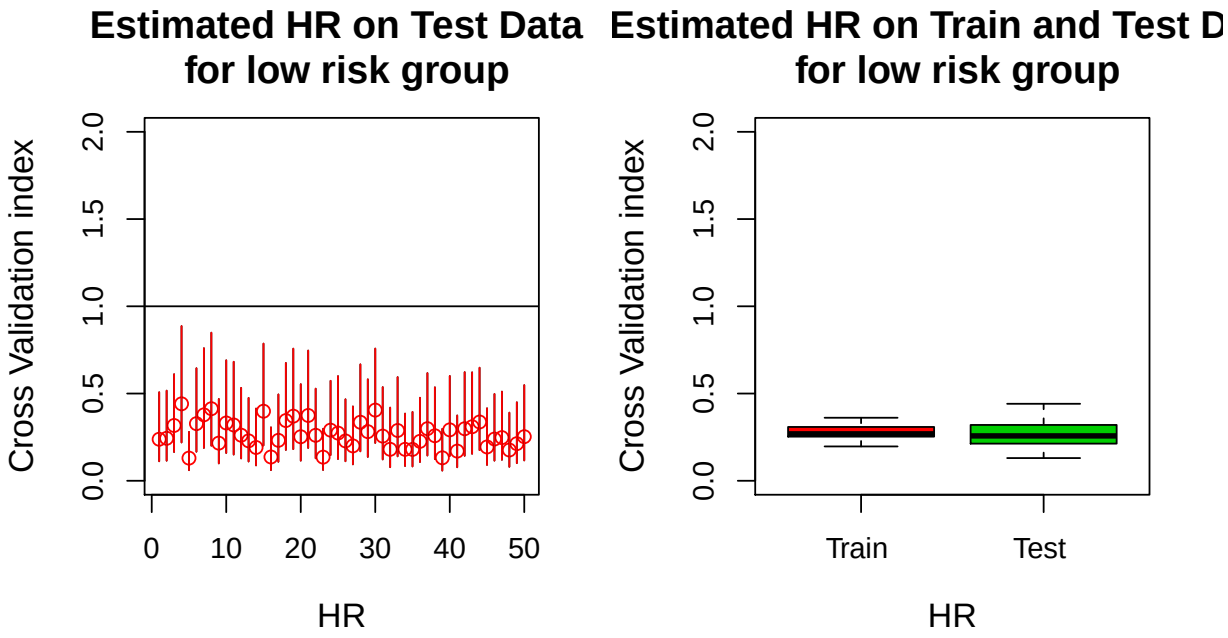

Figure 7: Cross-validated majority votes.

```
plot(CVR1, Which = 20, ylim = c(0, 5))
```

Mostly selected gene during the cross validations

The function `GeneOccurence()` searches for mostly selected genes during the cross-validation of gene by gene analysis. Top genes are ranked based on estimated HR for low risk. Therefore top gene should have minimum HR estimate. Number of top K genes to be considered can be given in advanced. Finally, function visualizes the genes that are selected at least 5% times during the cross-validations.

```
res <- GeneOccurence(CVR1, TopK = 20, minFreq = 5)
```

Cross validations for top K gene analysis

The function `CVSeqIncreaseGenes()` further processes the cross-validated gene by gene analysis results. The function sequentially considers top $K_1, K_2, \ldots, K_n$ number of genes based on the estimated HR on the test data of the cross-validated gene by gene analysis. For each top K genes, function recomputes first PCA or PLS on training data and estimates risk scores on both test and training data only on the gene expression matrix with top K genes. Patients are then classified as having low or high risk based on the test data where the cutoff used is median of the risk score based on training data. Finally, hazard ratio is estimated based on test data. The process is repeated for each top K gene sets.

The function can be interpreted as cross-validated version of the function `SeqIncreaseGenes()`.

```
CVTopK <- CVSeqIncreaseGenes(CVR1, top = seq(5, 100, by=5), SurvTime, Censor, ProgFact = NULL)

plot(CVTopK)

summary(CVTopK)
```

Cross validations for Lasso and ElasticNet

The function `CVLassoElasticNetCoxPh()` performs the cross-validations for LASSO and Elastic net models for Cox proportional hazard model. Top genes selected are being updated at each iteration and use in the classifier. That means predictive gene signature is varied iteration to iteration. The underline idea is to investigate the HR under training and test data as well as mostly selected genes during this process.

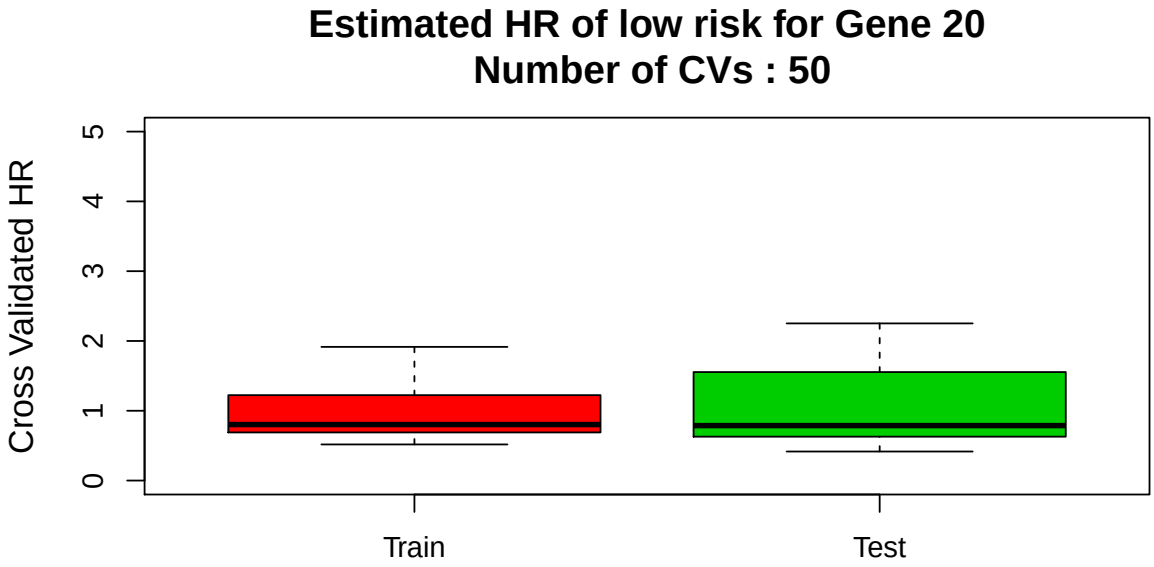

Figure 8: Cross-validated gene by gene analysis.

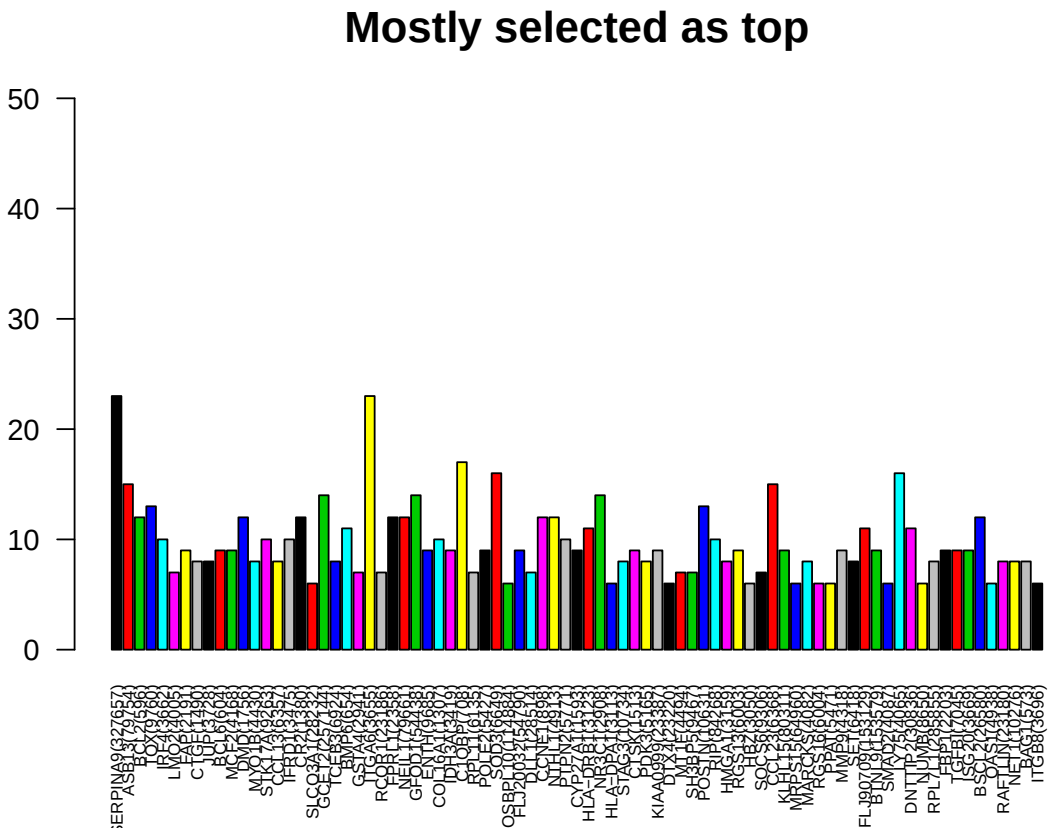

Figure 9: Mostly selected genes during cross-validated gene by gene analysis.

```
set.seed(123)
CVres <- CVLassoElasticNetCoxPh(n.cv = 50, fold = 3, SurvTime, Censor, Gdata,
  NuFeToBeSel = 100, ProgFact = NULL,
        ReduceDim = TRUE, GeneList = NULL,  StZ = TRUE, alpha = 1)

show(CVres)

Cross-validated Results for Lasso and Elastic Net based Predictive Gene signature
Number of CV:  50

summary(CVres)

Summary of Cross-validations
Estimated  quantiles of HR on test data
       5%       25%       50%       75%       95%
0.1422408 0.2001505 0.2567402 0.3218236 0.5270070

Estimated quantiles of HR on train data
        5%        25%        50%        75%        95%
0.03592761 0.04958284 0.07090524 0.08620618 0.10516886
Mostly selected 30 features:
 [1] "TPRT(23590)"      "COL19A1(1310)"    "TOPBP1(11073)"    "BMP6(654)"
 [5] "DMD(1756)"        "LILRA4(23547)"    "NR3C1(2908)"      "IFRD1(3475)"
 [9] "PRKD1(5587)"      "C1QBP(708)"       "RPL11(6135)"      "MCF2(4168)"
[13] "ITGA6(3655)"      "COL16A1(1307)"    "SFXN4(119559)"    "HBS1L(10767)"
[17] "MUS81(80198)"     "SLCO3A1(28232)"   "CCL18(6362)"      "GRK4(2868)"
[21] "HIAT1(64645)"     "PASK(23178)"      "SMAD3(4088)"      "DLL1(28514)"
[25] "HRK(8739)"        "GCET2(257144)"    "RYBP(23429)"      "POLE2(5427)"
[29] "CTGF(1490)"       "MGC61571(152100)"


plot(CVres, ptype = 1)


# Mostly selected 30 genes
plot(CVres, ptype = 3)
```

The same function can be used to perform cross-validations for Elastic net. We give an example of Elastic net with $\alpha = 0.2$.

```
set.seed(123)
ElNet <- CVLassoElasticNetCoxPh(n.cv = 20, fold = 3, SurvTime,
   Censor, Gdata, NuFeToBeSel = 150, ProgFact = NULL,
   ReduceDim = TRUE, GeneList = NULL,  StZ = TRUE, alpha = 0.2)
```

We further illustrate how we can use this function over fine grid of mixing parameter α while performing the cross-validations.

```
grid.alpha <- seq(0.1, 0.9, by = 0.1)

set.seed(123)
CvElNetResults <- sapply(grid.alpha,
  function(alpha) CVLassoElasticNetCoxPh(n.cv = 50, fold = 3, SurvTime,
                             Censor, Gdata, NuFeToBeSel = 100, ProgFact = NULL,
                              ReduceDim = TRUE, GeneList = NULL,  StZ = TRUE,
                              alpha = alpha))

#extract slots from S4 Class objects
Runtime <- sapply(1:length(grid.alpha),
  function(k) slot(CvElNetResults[[k]], "Run.Time"))
```

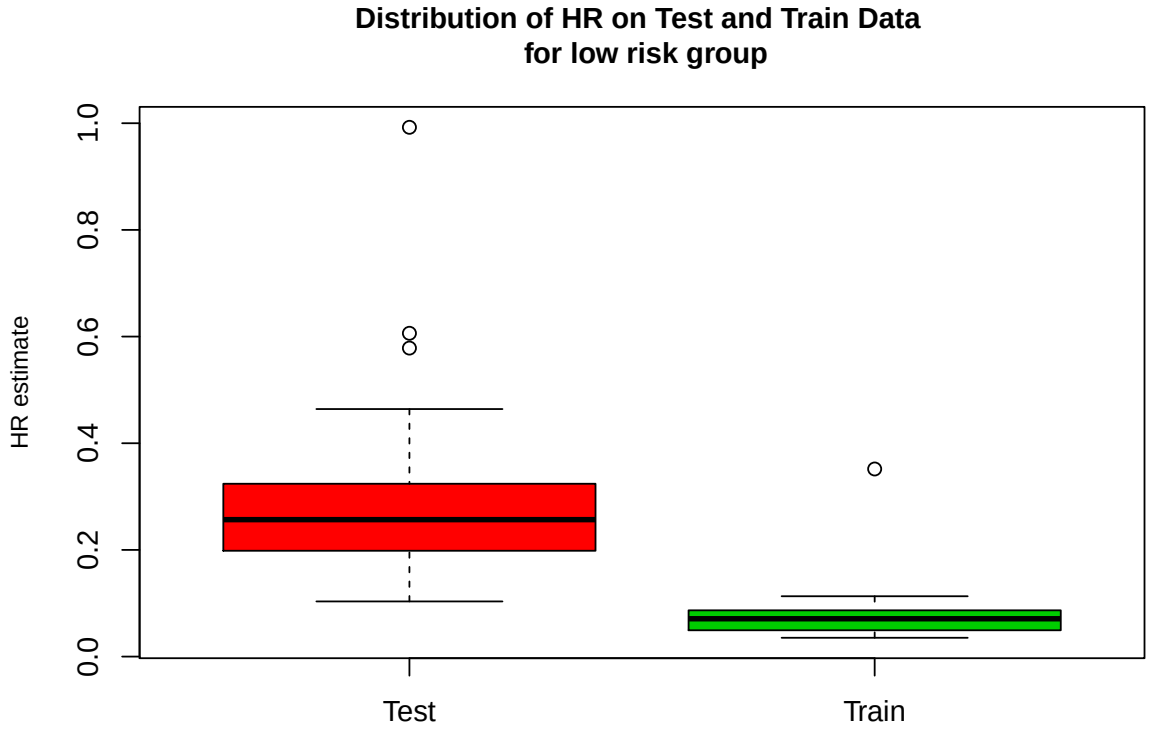

Figure 10: Cross-validated LASSO.

```
lambda <- sapply(1:length(grid.alpha),
  function(k) slot(CvElNetResults[[k]], "lambda"))
n.g    <- sapply(1:length(grid.alpha),
  function(k) slot(CvElNetResults[[k]], "n.g"))
HRT    <- sapply(1:length(grid.alpha),
  function(k) slot(CvElNetResults[[k]], "HRT")[, 1] )
HRTE   <- sapply(1:length(grid.alpha),
  function(k) slot(CvElNetResults[[k]], "HRTE")[, 1] )
HRTm   <- t(sapply(1:length(grid.alpha),
  function(k) quantile(slot(CvElNetResults[[k]], "HRT")[, 1],
          na.rm=T,probs = c(0.025, 0.5, 0.975))))
HRTEm  <- t(sapply(1:length(grid.alpha),
  function(k) quantile(slot(CvElNetResults[[k]], "HRTE")[, 1],
          na.rm=T,probs = c(0.025, 0.5, 0.975))))
freq   <- sapply(1:length(grid.alpha),
  function(k) colSums(slot(CvElNetResults[[k]], "gene.mat")))
```

```
colnames(HRTE) <- grid.alpha
boxplot(HRTE, ylim = c(0, max(HRTE)), names = grid.alpha[1:length(grid.alpha)],
 main = "HR on Testing Set 100 runs", col = 2:(length(grid.alpha)+1), ylab = "HR",
 xlab = expression(alpha), cex.main = 1.5, cex.lab = 1.3)
```

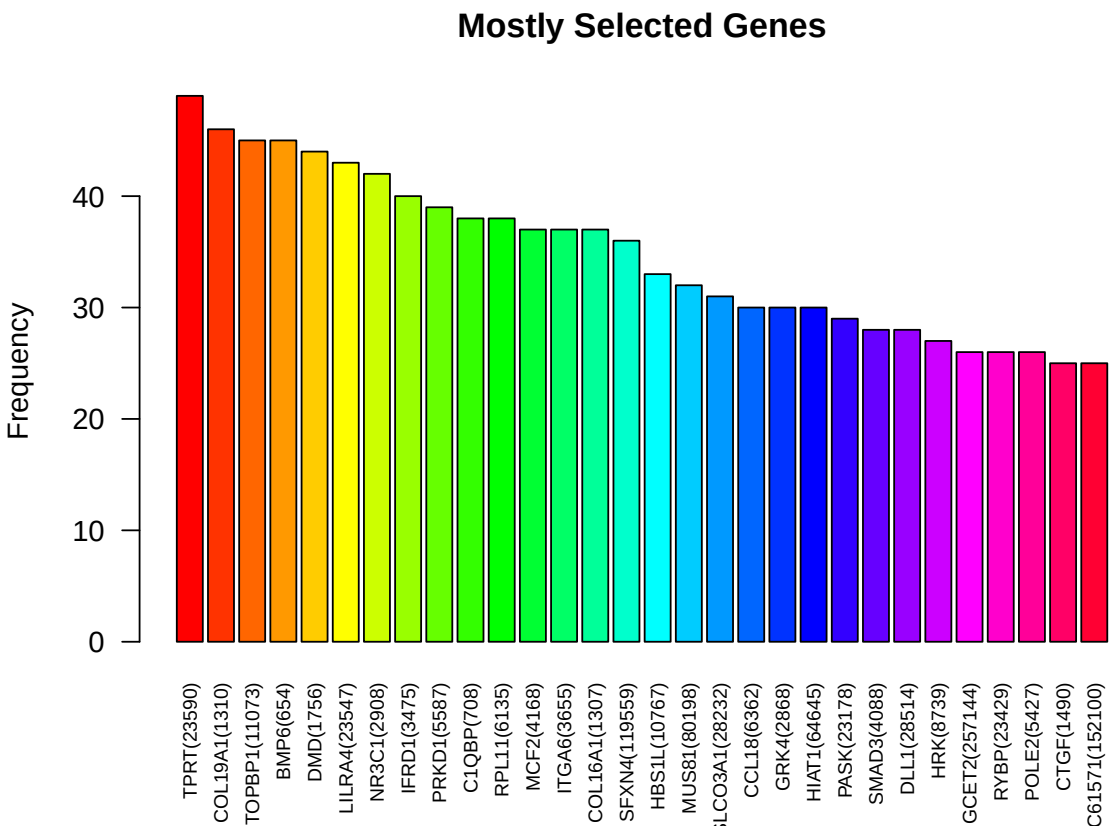

Figure 11: Cross-validated LASSO.

Inner and Outer Cross validations for Lasso and ElasticNet

Finally, we present the function `InnerCrossValELNet()` which can be used to perform inner and outer cross-validations for LASSO and Elastic net based models. Further cross-validations are run based on fixed gene list while classifier is evaluated on completely independent samples. The classifier is built on the weights obtained from the inner cross-validations results and it is tested on out of bag data. These weights can be fixed or can be updated at each outer iteration. If weights are not fixed then patients are classified using majority votes. Otherwise, weights obtained from the inner cross-validations are summarized by mean weights and used in the classifier. Inner cross-validations are performed by calling the function `CVLassoElasticNetCoxPh()`. Hazard ratio for low risk group is estimated using out-of-bag data.

```
#Select Top K genes (Mostly selected genes during the CV)
TopGenes <- c("TPRT(23590)", "COL19A1(1310)", "RPL7L1(285855)", "DMD(1756)", "NR3C1(2908)",
  "ITGA6(3655)", "PRKD1(5587)", "IFRD1(3475)", "BMP6(654)", "SFXN4(119559)", "TOPBP1(11073)",
  "LILRA4(23547)", "DNTTIP2(30836)", "HBS1L(10767)", "MUS81(80198)")


# without prognostic factors and weights are updated

WOpWup <- InnerCrossValELNet(fold = 3, n.cv = 20, n.innerCV = 50, MixParAlpha = 1,
                             Gdata, TopGenes, WeightsFixed = FALSE,
                             SurvTime, Censor, ProgFact = NULL)
show(WOpWup)
summary(WOpWup)

#Example II-----------------------------------------

# weights are fixed
```

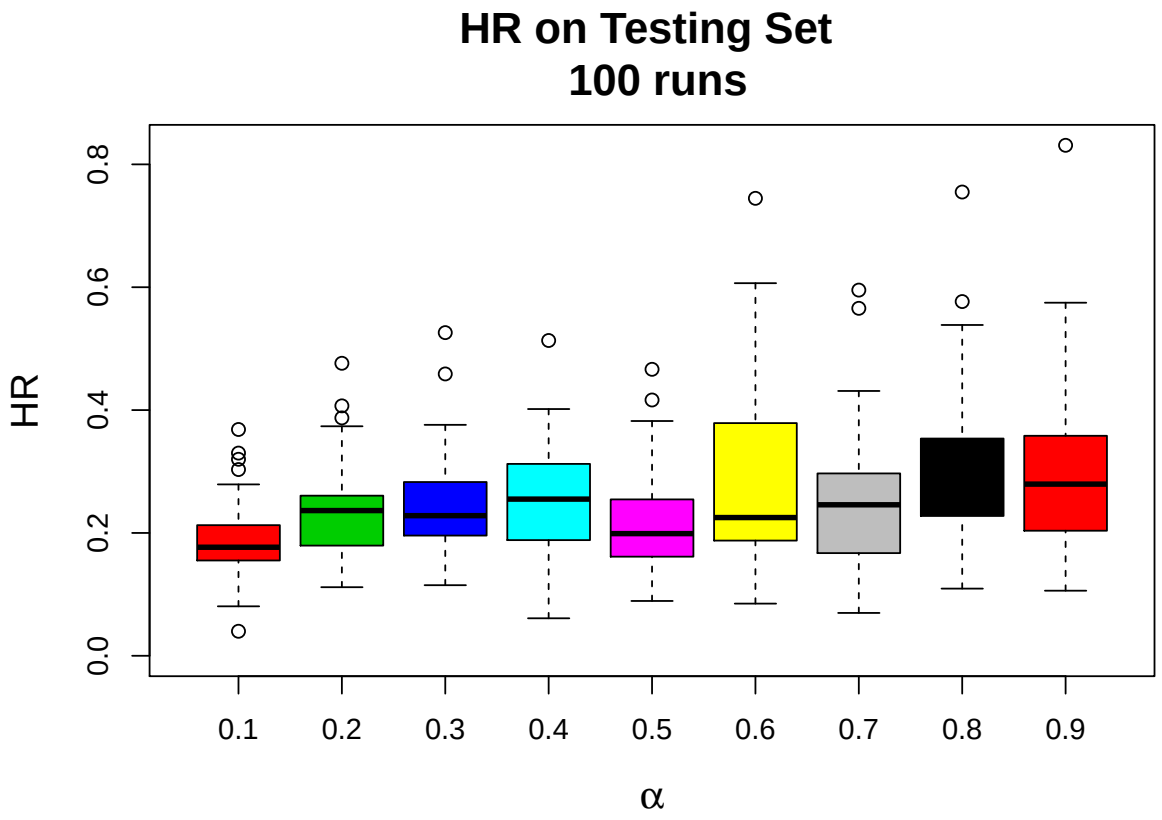

Figure 12: Cross-validated Elastic net.

```
WpWFtrue <- InnerCrossValELNet(fold = 3, n.cv = 20, n.innerCV = 50, MixParAlpha = 1,
                               Gdata, TopGenes, WeightsFixed = TRUE,
                               SurvTime, Censor, ProgFact = NULL)
show(WpWFtrue)
summary(WpWFtrue)

# weights are NOT fixed
WpWFfalse<-InnerCrossValELNet(fold = 3,n.c v= 20, n.innerCV = 50, MixParAlpha = 1,
                               Gdata, TopGenes, WeightsFixed = FALSE,
                               SurvTime, Censor, ProgFact)
show(WpWFfalse)
summary(WpWFfalse)

plot(WpWFtrue, ptype = 1)
```

Compare results based on classifier with fixed weights versus classifier with weights changing for which final classification is done by majority votes.

```
# estimated HR at each iteration for weights NOT fixed
Res <- data.frame(HRTrue = slot(WpWFtrue, "HRTE")[, 1], HRfalse = slot(WpWFfalse, "HRTE")[, 1])
boxplot(Res, col = "red", ylim = c(0, 1), main = "Estimated HR based on completely
    independent Samples", names = c("Weights Fixed", "Weights Updated"))
# estimated density of the HR
plot(WpWFtrue, ptype = 2)
```

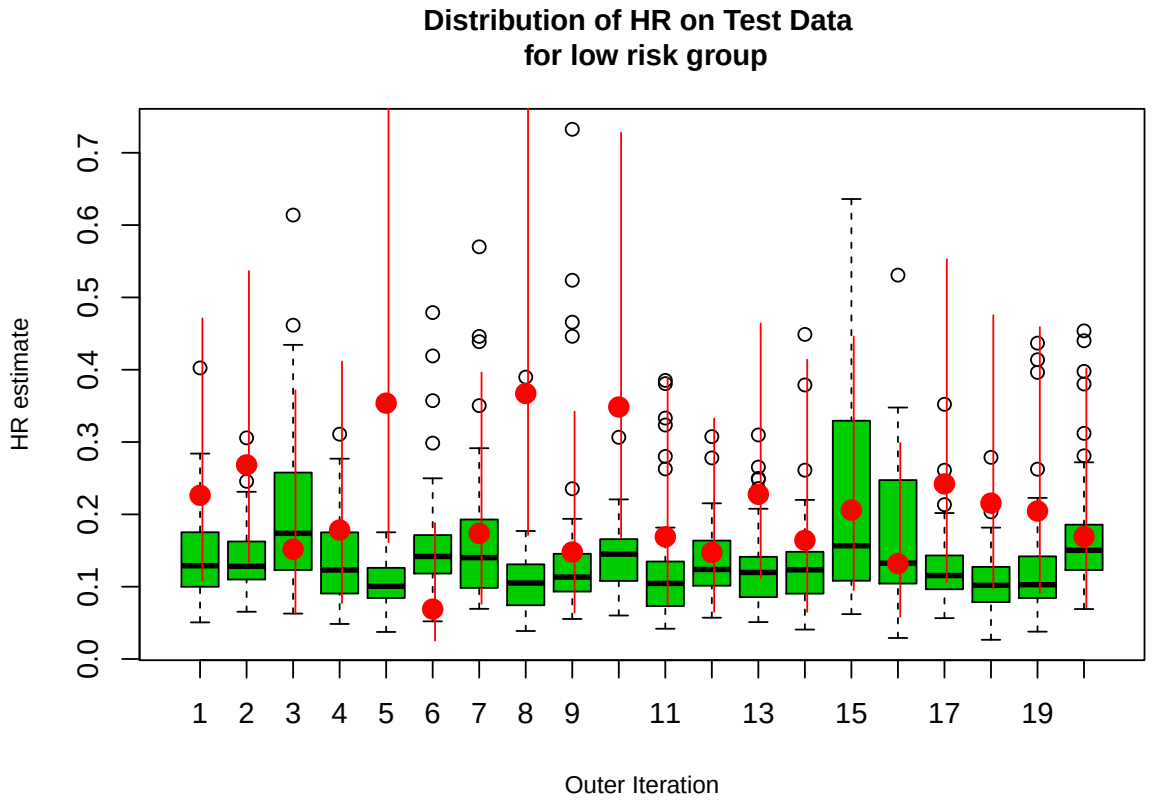

Figure 13: Cross-validated Elastic net.

Approximating the Null distribution of the HR

Another interesting function available in this package is `NullDistHR()` generates the null distribution of the HR by permuting the observations. Several ways of permutation setting are implemented. Function can be used to generate null distributions for four different validation schemes, PLS, PCA, Majority votes and LASSO based. Figure 14 shows the perumation results based on the Lasso analysis scheme. Note that this function internally calls functions: `SurFitPcaClasif`, `SurFitPlsClasif`, `MajorityVotes`, and `LassoElasticNetCoxPh`.

```
set.seed(123)
Perm<-NullDistHR(n.perm=50,case=2, Validation="Lasso", SurvTime, Gdata,
                         Censor,  ReduceDim=TRUE,  NuFeToBeSel=150,
                         ProgFact=NULL,  MedianCut = NULL )
show(Perm)

summary(Perm)

plot(Perm)
```

Summary

This package is useful in discovering and validating predictive gene signature for classifying patients into low risk or high risk in early phase clinical trials. For this particular package, the primary end point is survival, and classification of patients into low risk or high risk groups is mainly based on median risk score cutoff, but others can be considered as well. It can also accommodate the prognostic factors if any. Moreover treatment information can also be incorporated if available. Both statistical and machine learning techniques are integrated as validating suit. The package can be used to perform the analysis using the entire samples and can also be used to perform large scale cross-validations. For the first instance, package reduces larger gene

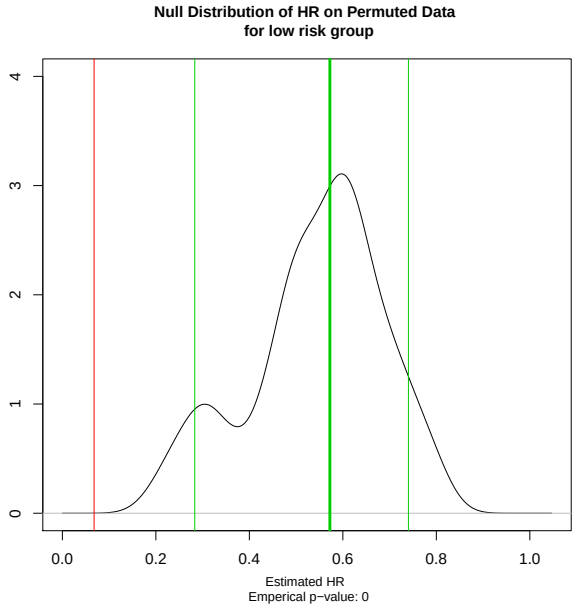

Figure 14: Estimated null distribution of HR.

expression matrix to smaller version using supervised principle components analysis. Later entire validation procedure can be performed using reduced gene expression matrix with various types of validation schemes. In conclusions package integrates the exiting methods which are well known techniques and R functions and packages as validation suit for biomakers and increase useability, visualizations and scope of the validation of biomakers specially in lager p and small N settings.

Bibliography

E. Bair and R. Tibshirani. Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biol*, 2(4):e108, 04 2004. doi: 10.1371/journal.pbio.0020108. [p1]

E. Bair, T. Hastie, D. Paul, and R. Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2006. doi: 10.1198/016214505000000628. [p1, 2]

D. Beer, S. Kardia, and C. Huang. Gene-expression profiles predict survival of patients with lung adenocarcinoma. *Nat. Med.*, 8:816–824, 2002. [p1]

X. Chen, L. Wang, and J. Smith. Supervised principal component analysis for gene set enrichment of microarray data with continuous or survival outcomes. *Bioinformatics*, 24:2479–2481, 2008. [p1]

K. Devarajan, Y. Zhou, N. Chachra, and N. Ebrahimi. A supervised approach for predicting patient survival with gene expression data. In *BioInformatics and BioEngineering (BIBE), 2010 IEEE International Conference on*, pages 26–31, 2010. doi: 10.1109/BIBE.2010.14. [p2]

J. Gui and H. Li. Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, 21(13):3001–3008, 2005. doi: 10.1093/bioinformatics/bti422. [p1]

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer-Verlag, New York, 2001. [p3]

D. V. Nguyen and D. M. Rocke. Partial least squares proportional hazard regression for application to dna microarray survival data. *Bioinformatics*, 18(12):1625–1632, 2002. doi: 10.1093/bioinformatics/18.12.1625. [p1]

A. Rosenwald, G. Wright, W. C. Chan, J. M. Connors, E. Campo, R. I. Fisher, R. D. Gascoyne, H. K. Muller-Hermelink, E. B. Smeland, J. M. Giltnane, E. M. Hurt, H. Zhao, L. Averett, L. Yang, W. H. Wilson, E. S. Jaffe,

R. Simon, R. D. Klausner, J. Powell, P. L. Duffey, D. L. Longo, T. C. Greiner, D. D. Weisenburger, W. G. Sanger, B. J. Dave, J. C. Lynch, J. Vose, J. O. Armitage, E. Montserrat, A. López-Guillermo, T. M. Grogan, T. P. Miller, M. LeBlanc, G. Ott, S. Kvaloy, J. Delabie, H. Holte, P. Krajci, T. Stokke, and L. M. Staudt. The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *New England Journal of Medicine*, 346(25):1937–1947, 2002. doi: 10.1056/NEJMoa012914. URL http://www.nejm.org/doi/full/10.1056/NEJMoa012914. PMID: 12075054. [p5]

R. Simon. Development and evaluation of therapeutically relevant predictive classifiers using gene expression profiling. *J. Natl. Cancer Instit.*, pages 1169–1171, 98 2006. [p1]

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B Methodological*, 36(2):111–147, 1974. [p4]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. [p3]

H. Zhou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67(2):301–320, 2004. [p3]

Pushpike J. Thilakarathne
Interuniversity Institute for Biostatistics and Statistical
Bioinformatics, CenStat, Universiteit Hasselt,
Agoralaan 1, B3590 Diepenbeek, Belgium
pushpike.thilakarathne@uhasselt.be

Ziv Shkedy
Interuniversity Institute for Biostatistics and Statistical
Bioinformatics, CenStat, Universiteit Hasselt,
Agoralaan 1, B3590 Diepenbeek, Belgium
ziv.shkedy@uhasselt.be

Martin Otava
Interuniversity Institute for Biostatistics and Statistical
Bioinformatics, CenStat, Universiteit Hasselt,
Agoralaan 1, B3590 Diepenbeek, Belgium
martin.otava@uhasselt.be

Willem Talloen
Janssen, Pharmaceutical companies of Johnson & Johnson,
Turnhoutseweg 30, B-2340 Beerse, Belgium
wtalloen@its.jnj.com

Luc Bijnens
Janssen, Pharmaceutical companies of Johnson & Johnson,
Turnhoutseweg 30, B-2340 Beerse, Belgium
lbijnens@its.jnj.com

Adetayo Kasim
Wolfson Research Institute, Durham University,
University Boulevard, Thornaby, Stockton-on-Tees, TS17 6BH, United Kingdom
a.s.kasim@durham.ac.uk