Usage of `aBioMarVsuit` Package (version 1.0)

Pushpike Thilakarathne
I-BioStat,
University of Hesselt
Belgium

January 14, 2013

Contents

1 Introduction

This particular R package `aBioMarVsuit` stands for a biomarker validation suit for predicting survival out come.

This package is useful in finding and validating predictive gene signature for classifying low risk versus high risk patients in early phase clinical trials. The primary end point is survival, and classification of cancer patients into low risk or high risk groups is mainly based on median cutoff, but others can be considered as well. It can also accommodate the prognostic factors if any. 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 carryout large scale cross validations. For the first instance, package reduces larger gene 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.

This vignette is organized as follows. First, we will introduce several functions that can be used to analyze entire datasets. In the later part of this document we will show more advance functions that can be used to perform various type of cross validation schemes.

2 Load the Data

First we load the package `aBioMarVsuit` and then well known real-life data set `DLBCL`. The dataset contains an expression set on diffuse large B-cell lymphoma (for more details see Rosenwald et al̇2002).

```
R> library(aBioMarVsuit)
```

2.1 DLBCL data set

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

[1] 3583  179

R> SurvTime<-SurvData[!is.na(SurvData[,c("IPI")]),c("FollowUpYears")]
R> Censor<-ifelse(SurvData[!is.na(SurvData[,c("IPI")]),c("StatusAtFollowUp")]=="Dead",1,0)
R> Gdata[is.na(Gdata)]<-mean(Gdata,na.rm=T)
R> #generate some prognostic factors
R> nPatients<-ncol(Gdata)
R> ProgFact<-data.frame(Age=floor(SurvTime*0.68+rnorm(nPatients,30,10)),
                  Stage=sample(1:4,nPatients,replace=T),sex=rbinom(nPatients, 1, 0.5))
```

2.2 Simulated Data

Alternatively, simulated data sets can also be very valuable. To that end we have developed a new function that can generate some synthetic survival data and gene expression data.

The function generates the gene expression matrix where small set of genes (50) are informative and rest of them are set as noisy genes. Next to that Survival time and Censoring information are generated based on first right singular vectors of svd of the gene expression matrix. It also generates other prognostic factors such as Age, Stage and sex which slightly correlated with survival time.

```
R> SimData<-GenSynSurvData(nPatients=100,nGenes=150,Pi=0.5)
R> SurvTime<-SimData$SurvTime
R> Censor<-SimData$Censor
R> ProgFact<-SimData$ProgFact
R> Gdata<-SimData$Gdata
R> featurenames<-SimData$featurenames
```

3 Basic Functions

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.

3.1 Function EstimHR

The function does the classification based on the risk scores provided by the user and visualize survival fit along with HR estimate.

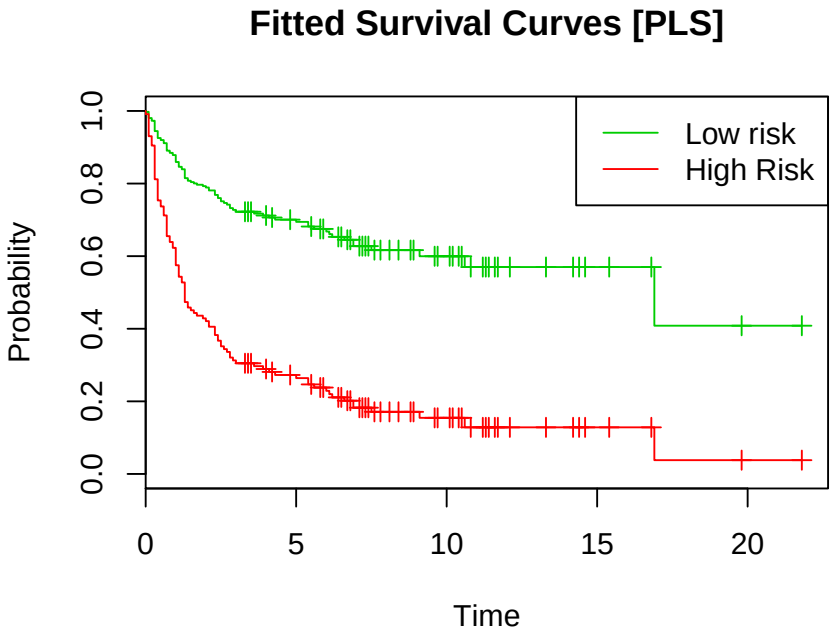
```
R> EstimHR(RiskScore, Sdata, ProgFact = NULL, Plots = FALSE, MedianCut = NULL)
```

3.2 Function SurFitPlsClasif

This function reduces larger gene expression matrix to smaller version using supervised pca approach. The function performs the PLS on reduced gene expression matrix and fit Cox proportional hazard model with first PLS scores as a covariate afterwards. And classifier is then 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 cut off values and default is median cut off. This function can handle single and multiple genes. Other prognostic factors can be included to the model.

```
R> results1<-SurFitPlsClasif(SurvTime, Gdata,  Censor, ReduceDim=TRUE, NuFeToBeSel=100,
                  ProgFact=NULL, Plots = FALSE,MedianCut = NULL )
R> #Estimated HR for low risk group
R> summary(results1$SurFit)[[8]][-2]

[1] 0.2738225 0.1790867 0.4186728

R>

R> plot(survfit(results1$SurFit,newdata=data.frame(gid=as.factor(c("low","high")))),col=3:2,
                  main="Fitted Survival Curves [PLS]",xlab="Time",ylab="Probability")
R> legend("topright",c("Low risk","High Risk"),col=3:2,lty=c(1,1))
```



3.3 Function SurFitPcaClasif

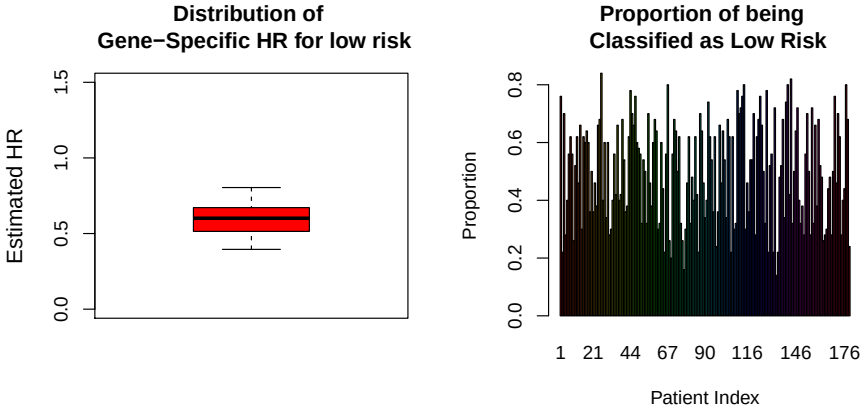
This works more or less the same way as function SurFitPlsClasif but function uses first PCA scores on the reduced gene expression matrix to build up the classifier.

```
R> results1<-SurFitPcaClasif(SurvTime, Gdata, Censor, ReduceDim=TRUE, NuFeToBeSel=100,
                          ProgFact=NULL,  Plots = FALSE,      MedianCut = NULL )
R> summary(results1$SurFit)
```

3.4 Function GeneSpecificCoxPh

This 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 levels.

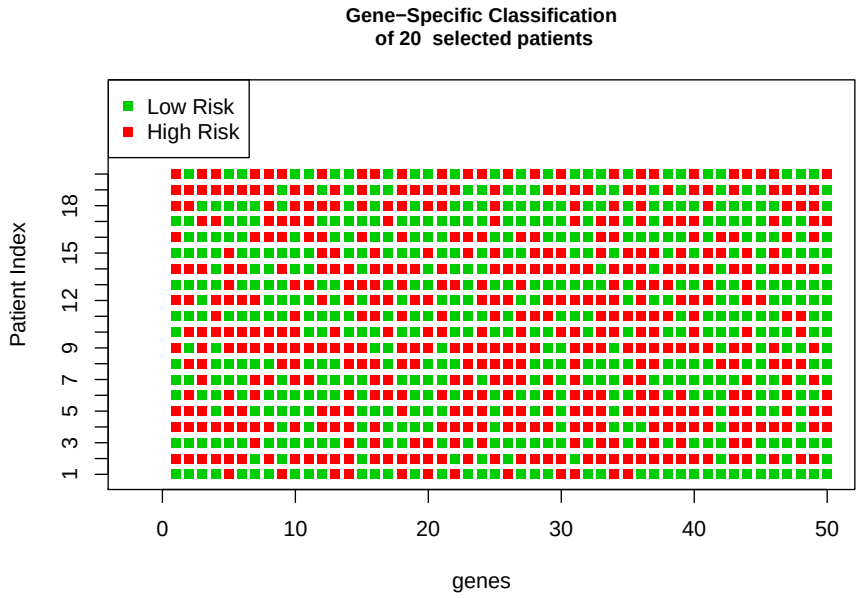
```
R> Ana1<-GeneSpecificCoxPh( SurvTime, Gdata, Censor,  ReduceDim=TRUE, NuFeToBeSel=50,
                          ProgFact=NULL,  MedianCut = NULL)
R> show(Ana1)

Gene by Gene CoxPh Model
Number of Genes used:  50

R> 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

R> plot(Ana1)
```



3.5 Function MajorityVotes

The function 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.

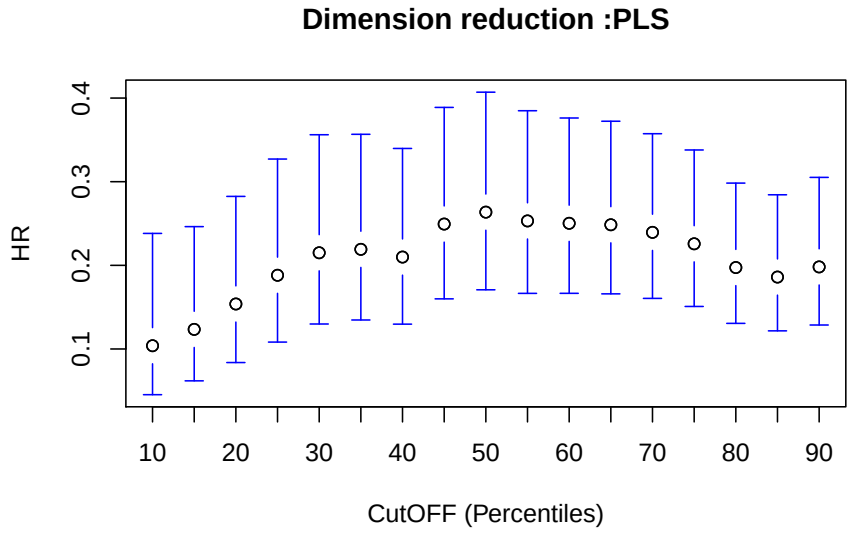
```
R> MVres<-MajorityVotes(Ana1,ProgFact=NULL, SurvTime,Censor,J=1)
```



3.6 Function GridAnalysis

The function performs the sensitivity of the cut off values used in classification and percentiles of risk score obtained under SurFitPlsClasif or SurFitPcaClasif methods is used as fine grid points.

```
R> 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
```



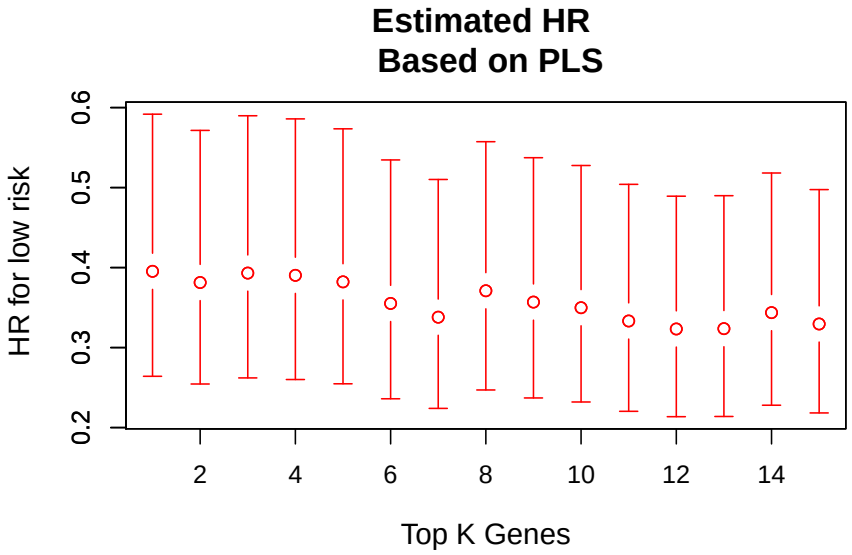
Other dimension reduction methods can also considered.

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

3.7 Function SeqIncreaseGenes

This function sequentially increases the number of top K genes to be used in the PCA or PLS methods in order to obtain the risk score.

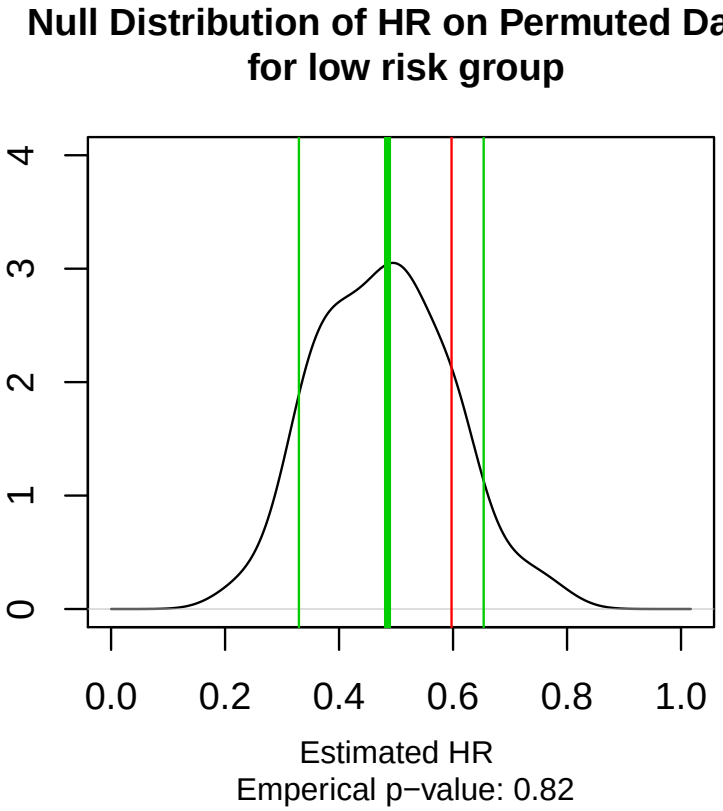
```
R> data(exprLym)
R> GexprMatrix<-exprs(exprLym)
R> SurvData<-pData(exprLym)
R> PatietId<-rownames(SurvData[!is.na(SurvData[,c("IPI")]),])
R> Gdata<-GexprMatrix[,PatietId]
R> SurvTime<-SurvData[!is.na(SurvData[,c("IPI")]),c("FollowUpYears")]
R> Censor<-ifelse(SurvData[!is.na(SurvData[,c("IPI")]),c("StatusAtFollowUp")]=="Dead",1,0)
R> Gdata[is.na(Gdata)]<-mean(Gdata,na.rm=T)

R> 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
 [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
```



3.8 Function NullDistHR

This function generates the null distribution of the HR by permuting the observations. Several ways of permutation setting are implemented. That is, function can be used to generate null distributions for four different validation schemes, PLS based, PCA based, Majority votes based and Lasso based.

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

Estimated Null Ditribution of the  PLSbased
Number of Permutations:  50

R> summary(Perm)

Summary of Permutation Analysis
validation scheme used : PLSbased
Number of Permutations:  50
Estimated  quantiles of the null distribution of HR
       5%       25%       50%       75%       95%
0.3295615 0.3901207 0.4850075 0.5611891 0.6537271

Estimated HR on original data
 Estimate lower95CI Upper95CI
0.5973303 0.4028807 0.8856306

R> plot(Perm)
```



3.9 Function LassoElasticNetCoxPh

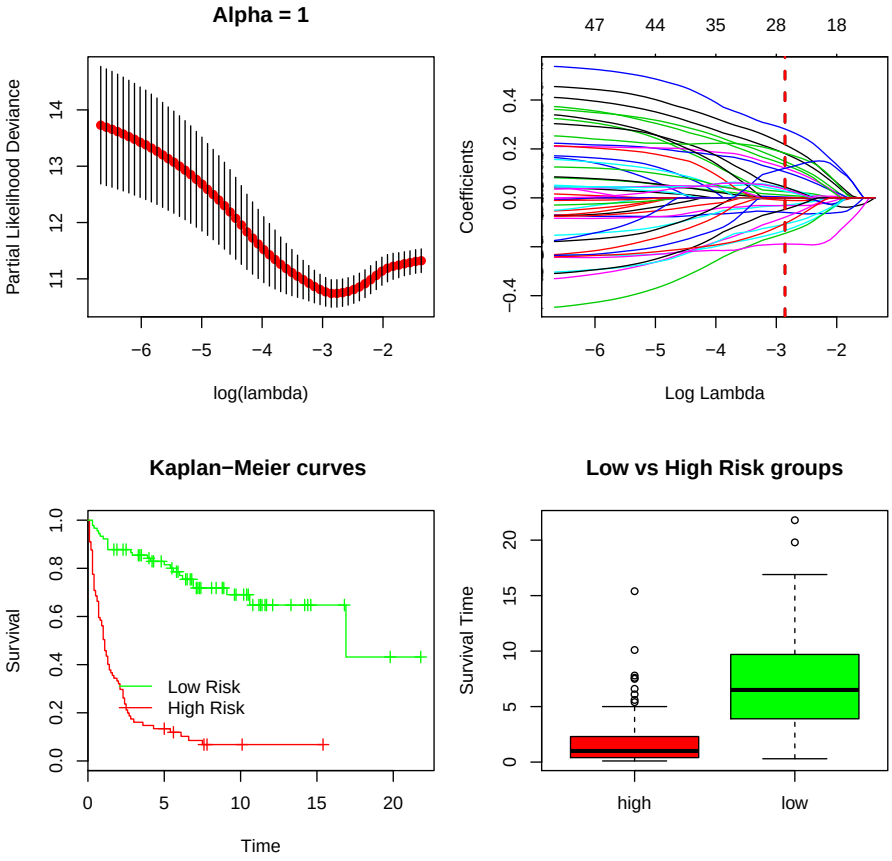
This is a wrapper function for glmnet and fit model with all prognostic factors and genes. LASSO, Elastic net and Ridge regressions can be fitted. Optimum lambda will be used to select the non-zero shrinkage coefficients. The function can accommodates prognostic factors and they will be forced to be in the model if they are supplied. This returns estimated HR for low risk group and class labels for each patients and other interesting objects.

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

Application of Lasso with 50 genes

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



4 More Advance Functions.

In this we focus on more advanced functions that can be used for cross validations.

4.1 CVforDimPcaPls

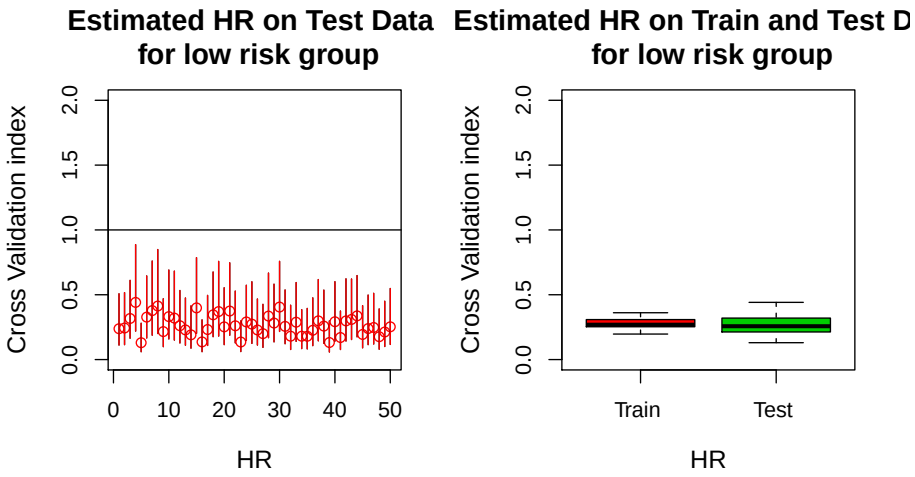
Cross validations for the analysis performs by SurFitPlsClasif and SurFitPcaClasif functions where the dimension reduction methods are PCA and PLS.

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

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

4.2 CvForMajorityVotes

Performs cross validations for Majority votes based classification.

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

R> show(cvmvres)

Cross Validated Majority Votes Based Classification Analysis
Number of cross valdiations used:  50

R> plot(cvmvres)
```



4.3 CVGeneSpecificCoxPh

This function performs the cross validation for gene by gene analysis. First data will be divided into train and test. Second, gene-specific model is fitted on train data and classifier is built. And classifier is then evaluated on test data for that particular gene. Process is repeated for all genes. All these steps can be repeated many times.

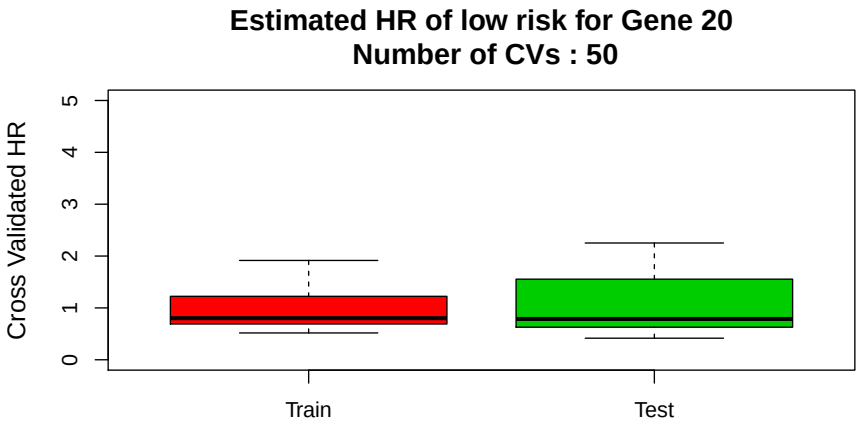
```
R> CVR1<-CVGeneSpecificCoxPh(fold=3, SurvTime, Gdata, Censor,
                          ReduceDim=TRUE,  NuFeToBeSel=50,
                          ProgFact=NULL, MedianCut = NULL,
                          n.cv=50)

R> show(CVR1)
```

```
Cross Validated Gene by Gene Analysis
Number of Genes used:  150
Number of cross valdiations used:  50
```

```
R> 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)
```

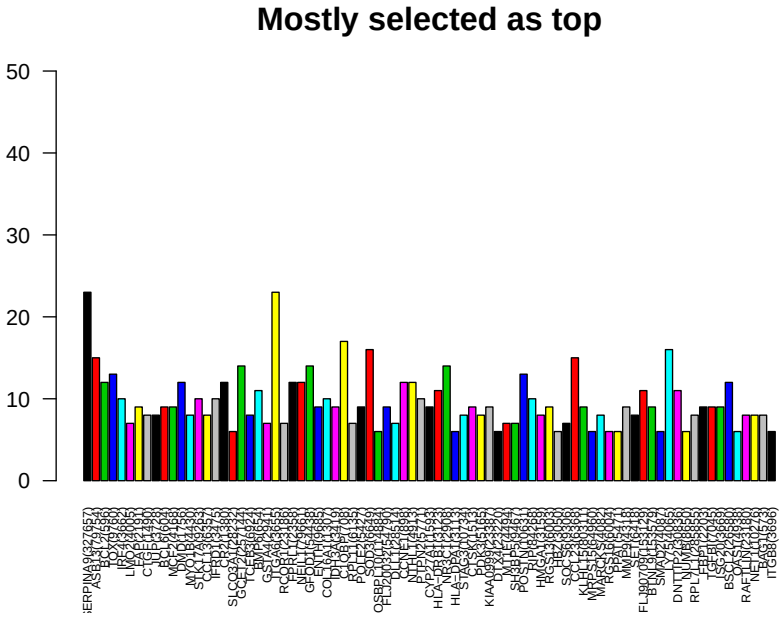
```
R> plot(CVR1, Which=20,ylim=c(0,5))
```



4.4 GeneOccurence

This function searches for mostly selected top 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 need to be considered can be given in advanced. Finally it visualizes the genes that are selected at least 5% times during the cross validations.

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



4.5 GeneOccurence

This function further processes the cross validated gene by gene analysis results. And function sequentially considers top $k_1, k_2, ..., k_n$ 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 recompute first PCA or PLS on train data and estimate risk scores on both test and train 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 train data. Finally hazard ratio is estimated based on test data. The process is repeated for each top K gene sets.

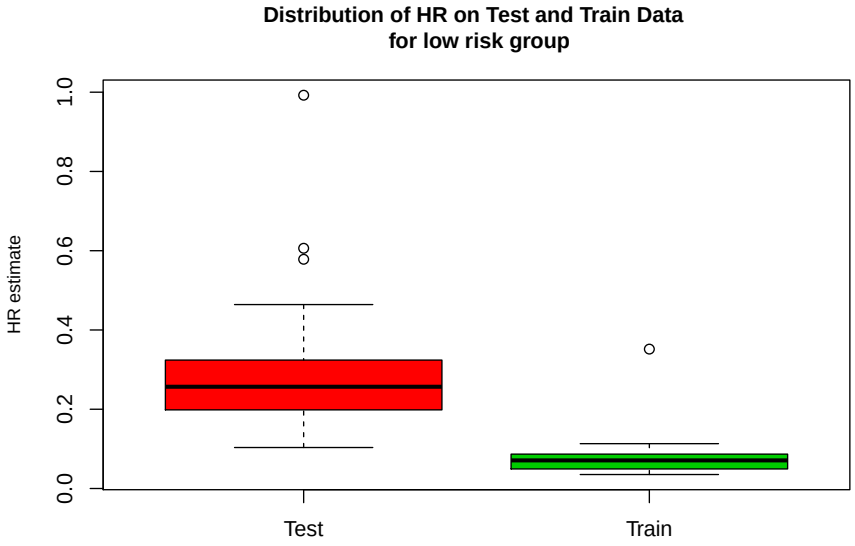
The function can be interpreted as cross validated version of the function SeqIncreaseGenes.

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

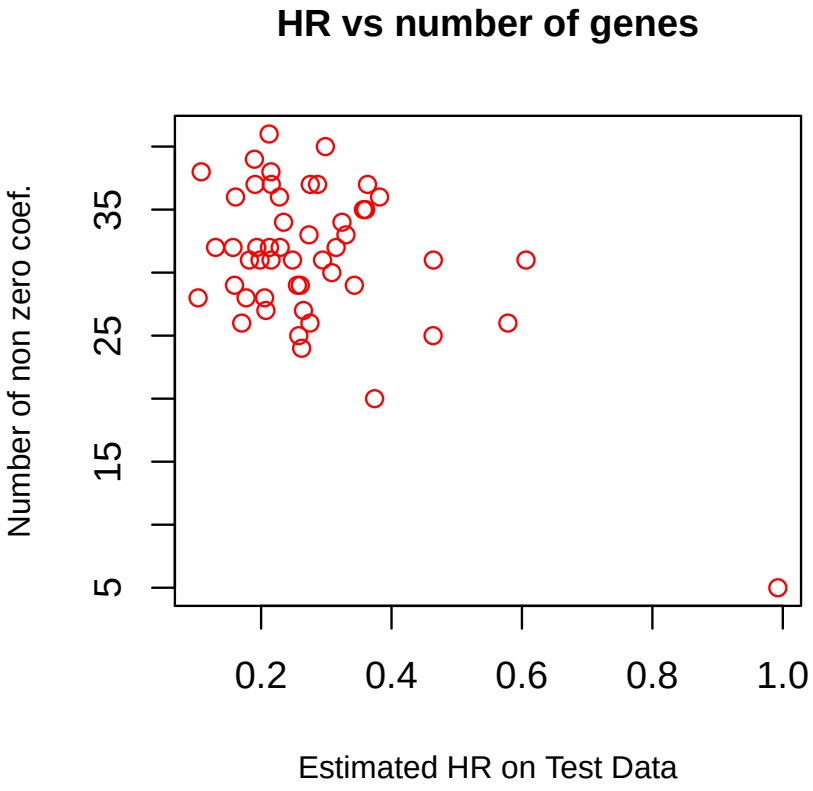
4.6 CVLassoElasticNetCoxPh

The function 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 train and test data as well as mostly selected genes during this process.

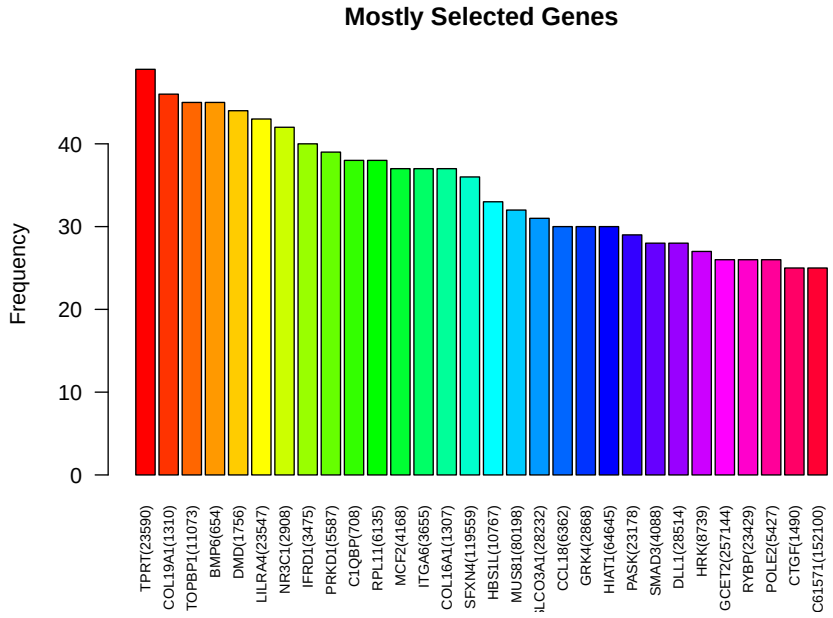
```
R> #Without Prognostic Factors
R> set.seed(123)
R> CVres<-CVLassoElasticNetCoxPh(n.cv=50,fold=3,SurvTime, Censor, Gdata,
                 NuFeToBeSel=100, ProgFact=NULL,
        ReduceDim=TRUE, GeneList=NULL,  StZ=TRUE, alpha=1)
R>

R> show(CVres)

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

R> 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)"

R> #distribution of the HR
R> plot(CVres,ptype=1)
```



```
R> #estimated HR vs number of genes selected
R> plot(CVres,ptype=2)
```




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



4.6.1 Cross Validation for Elastic Net

Cross validations for Elastic Net with $\alpha = 0.2$.

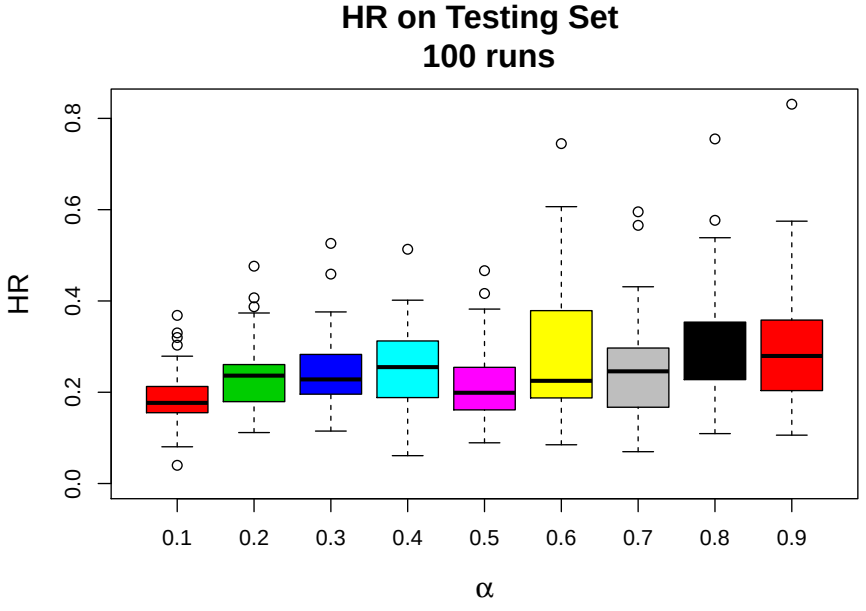
```
R> set.seed(123)
R> ElNet<-CVLassoElasticNetCoxPh(n.cv=20,fold=3,SurvTime,
   Censor, Gdata, NuFeToBeSel=150, ProgFact=NULL,
   ReduceDim=TRUE, GeneList=NULL,  StZ=TRUE, alpha=0.2)
```

4.6.2 Apply Elastic Net on grid of α values and cross validations

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

```
R> set.seed(123)
R> 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))
R> #extract slots from S4 Class objects
R> Runtime<-sapply(1:length(grid.alpha),
                  function(k) slot(CvElNetResults[[k]], "Run.Time"))
R> lambda <-sapply(1:length(grid.alpha),
                  function(k) slot(CvElNetResults[[k]], "lambda"))
R> n.g    <-sapply(1:length(grid.alpha),
                  function(k) slot(CvElNetResults[[k]], "n.g"))
R> HRT    <-sapply(1:length(grid.alpha),
                  function(k) slot(CvElNetResults[[k]], "HRT")[,1] )
R> HRTE   <-sapply(1:length(grid.alpha),
                  function(k) slot(CvElNetResults[[k]], "HRTE")[,1] )
R> 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))))
R> 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))))
R> freq   <-sapply(1:length(grid.alpha),
                  function(k) colSums(slot(CvElNetResults[[k]], "gene.mat")))
```

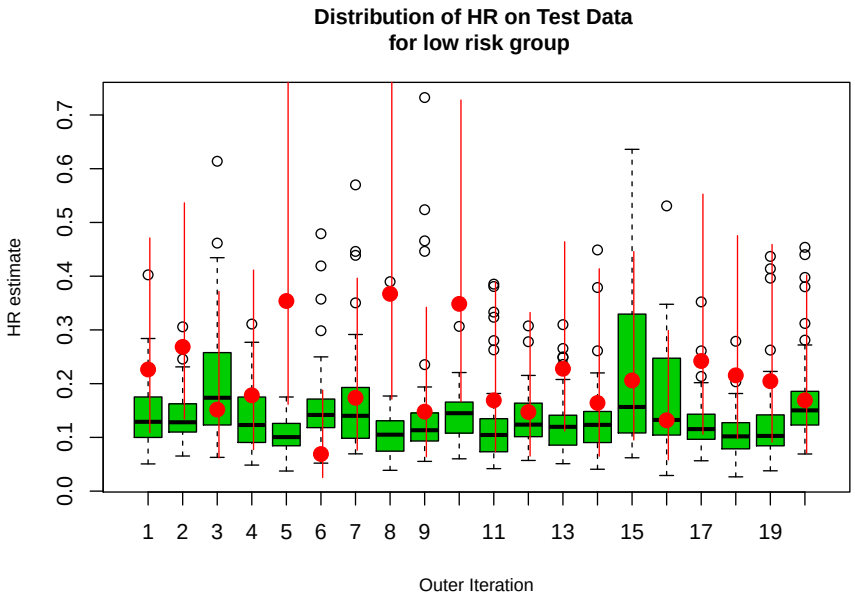
```
R> colnames(HRTE)<-grid.alpha
R> boxplot(HRTE,ylim=c(0,max(HRTE)),names=grid.alpha[1:length(grid.alpha)],
 main="HR on Testing Set \n 100 runs",col=2:(length(grid.alpha)+1),ylab="HR",
 xlab=expression(alpha),cex.main=1.5,cex.lab=1.3)
```



4.7 InnerCrossValELNet

The function can be used to perform the further cross validations based on fixed gene list while classifier is evaluated on completely independent samples. The classifier is built on the weights obtain 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 to function CVLassoElasticNetCoxPh. Hazard ratio for low risk group is estimated using out-of-bag data.

```
R> #Select Top K genes (Mostly selected genes during the CV)
R>  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)")
R> #Example I---------------------------------------
R> # without prognostic factors and Weights are updated
R>
R> WOpWup<-InnerCrossValELNet(fold=3,n.cv=20,n.innerCV=50,MixParAlpha=1,
                              Gdata,TopGenes,WeightsFixed=FALSE,
                              SurvTime, Censor,ProgFact=NULL)
R> show(WOpWup)
R> summary(WOpWup)
R> #Example II----------------------------------------
R> #with prognostic factors
R> #and Weights are fixed
R>
R> WpWFtrue<-InnerCrossValELNet(fold=3,n.cv=20,n.innerCV=50,MixParAlpha=1,
                              Gdata,TopGenes,WeightsFixed=TRUE,
                              SurvTime, Censor,ProgFact)
R> show(WpWFtrue)
R> summary(WpWFtrue)
R> #and Weights are NOT fixed
R> WpWFfalse<-InnerCrossValELNet(fold=3,n.cv=20,n.innerCV=50,MixParAlpha=1,
                              Gdata,TopGenes,WeightsFixed=FALSE,
                              SurvTime, Censor,ProgFact)
R> show(WpWFfalse)
R> summary(WpWFfalse)

R> plot(WpWFtrue,ptype=1)
```




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