Package 'PPLasso'

February 8, 2022

Type Package

Title Prognostic Predictive Lasso for Biomarker Selection

Version 1.0

Date 2022-02-07

Author Wencan Zhu [aut, cre],
Celine Levy-Leduc [ctb],
Nils Ternes [ctb]

Maintainer Wencan Zhu <wencan.zhu@agroparistech.fr>

Description We provide new tools for the identification of prognostic and predictive biomarkers. For further details we refer the reader to the paper <arXiv:2202.01970> (Zhu et al., 2022).

License GPL-2

Imports genlasso, ggplot2, cvCovEst, glmnet, MASS

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2022-02-08 10:10:05 UTC

R topics documented:

PPLasso-package *Prognostic Predictive Lasso for Biomarker Selection*

Description

We provide new tools for the identification of prognostic and predictive biomarkers. For further details we refer the reader to the paper <arXiv:2202.01970> (Zhu et al., 2022).

Details

The DESCRIPTION file:

Package:	PPLasso
Type:	Package
Title:	Prognostic Predictive Lasso for Biomarker Selection
Version:	1.0
Date:	2022-02-07
Authors@R:	c(person("Wencan", "Zhu", email = "wencan.zhu@agroparistech.fr", role = c("aut", "cre")), person("Ce
Author:	Wencan Zhu [aut, cre], Celine Levy-Leduc [ctb], Nils Ternes [ctb]
Maintainer:	Wencan Zhu <wencan.zhu@agroparistech.fr>
Description:	We provide new tools for the identification of prognostic and predictive biomarkers. For further details v
License:	GPL-2
Imports:	genlasso, ggplot2, cvCovEst, glmnet, MASS
VignetteBuilder:	knitr
Suggests:	knitr, rmarkdown
NeedsCompilation:	no
Packaged:	2022-02-07 08:33:08 UTC; Wencan
Depends:	R (>= 3.5.0)

Index of help topics:

This package provide usufull tool for the identification of prognostics and predictive biomarkers.

Author(s)

Wencan Zhu [aut, cre], Celine Levy-Leduc [ctb], Nils Ternes [ctb]

Maintainer: Wencan Zhu <wencan.zhu@agroparistech.fr>

References

W. Zhu, C. Levy-Leduc, N. Ternes. "A variable selection approach for highly correlated predictors in high-dimensional genomic data". (2020)

`Correction1Vect` *Correction on two vectors*

Description

For the estimation of β in Zhu et al. (2022), this function keeps only the M largest values coefficientss set the others to 0.

Usage

```
Correction1Vect(X, Y, te = NULL, vector, top_grill. = c(1:length(vector)), delta = 0.95)
```

Arguments

`X`	Design matrix
`Y`	Response vector
`te`	treatment effects
`vector`	The vector on which we performe the thresholding
`top_grill.`	grill of the thresholding
`delta`	parameter δ in the thresholding

Value

This function returns the thresholded vector.

Author(s)

Wencan Zhu, Celine Levy-Leduc, Nils Ternes

Examples

```
vect_sample=sample(1:20,20)
X=t(sapply(c(1:10),FUN=function(x) rnorm(20)))
Y=rnorm(10)

Correction1Vect(X=X, Y=Y, vector=vect_sample)

## The function is currently defined as
function(X, Y, te=NULL, vector, top_grill.=c(1:length(vector)), delta=0.95){
```

```
 beta_interm <- sapply(top_grill., top, vect = vector)
beta_te <- rbind(rep(te[1],length(top_grill.)), rep(te[2],length(top_grill.)), beta_interm)
 yhat <- as.matrix(X %*% beta_te)
 residuals <- sweep(yhat, 1, Y)
 mse_final_top <- colMeans(residuals^2)
 ratio_mse <- c()
 for (k in 1:(length(top_grill.) - 1)) {
   ratio_mse[k] <- round(mse_final_top[k + 1]/mse_final_top[k],6)
 }
 top_ratio <- min(which(ratio_mse >= delta))
 if (is.infinite(top_ratio)) {
   opt_final_top <- length(vector)
 }
 else {
   opt_final_top <- top_grill.[top_ratio]
 }

 return(round(top(vect = vector, thresh = opt_final_top), 6))

}
```

`Correction2Vect` *Correction on two vectors*

Description

For the estimation of $\tilde{\beta}$ in Zhu et al. (2022), this function keeps only the K1 largest values of prognostic biomarkers coefficients and the k2 largest value of the presictive biomarkers coefficients and set the others to the smallest value among the k1 (k2) largest of prognostic (predictive part).

Usage

```
Correction2Vect(X, Y, te=NULL, vector_prog, vector_pred,
top_grill.=c(1:length(vector_prog)), delta=0.95, toZero=FALSE)
```

Arguments

`X`	Design matrix
`Y`	Response vector
`te`	treatment effects
`vector_prog`	Vector of prognostic biomarkers
`vector_pred`	Vector of predictive biomarkers
`top_grill.`	grill of the thresholding
`delta`	parameter δ in the thresholding
`toZero`	should the threshold to 0 or not

Value

This function returns the thresholded vector.

Author(s)

Wencan Zhu, Celine Levy-Leduc, Nils Ternes

Examples

```
x1=sample(1:10,10)
x2=sample(1:10,10)

X=t(sapply(c(1:10),FUN=function(x) rnorm(20)))
Y=rnorm(10)

Correction2Vect(X=X, Y=Y, vector_prog=x1, vector_pred=x2)

## The function is currently defined as
function(X, Y, te=NULL, vector_prog, vector_pred,
top_grill.=c(1:length(vector_prog)), delta=0.95, toZero=FALSE){

    if(toZero){
      matrix_top_fix <- sapply(top_grill., top, vect=vector_prog)
      matrix_top_opt <- sapply(top_grill., top, vect=vector_pred)
    } else {
      matrix_top_fix <- sapply(top_grill., top_thresh, vect=vector_prog)
      matrix_top_opt <- sapply(top_grill., top_thresh, vect=vector_pred)
    }


    opt_top_opt <- mse_fix <- c()
    for(j in 1:length(top_grill.)){
      fix_temp <- matrix_top_fix[,j]
      mse_temp <- c()
      yhat <- X%*%c(te, fix_temp, matrix_top_opt[,1])

      mse_temp[1] <- sum((Y-yhat)^2)
      for(m in 2:length(top_grill.)){
        opt_temp <- matrix_top_opt[,m]
        threshed_vect <- c(te, fix_temp, opt_temp)
        yhat <- X%*%threshed_vect
        mse_temp[m] <- sum((Y-yhat)^2)
        ratio_mse <- round(mse_temp[m]/mse_temp[m-1], 6)
        if(ratio_mse >= delta){
          opt_top_opt[j] <- top_grill.[m]
          mse_fix[j] <- mse_temp[m]
          break
        }
      }
      if(m==length(top_grill.)){
        opt_top_opt[j] <- top_grill.[m]
        mse_fix[j] <- mse_temp[m]
```

```
    }
    if(j==1){
      ratio_final <- 0
    } else {
      ratio_final <- mse_fix[j]/mse_fix[j-1]
    }
    if(ratio_final >= delta){
      opt_fix <- j
      opt_opt <- m
      break
    }
  }

  if(exists("opt_fix")==FALSE){
    opt_fix <- ncol(matrix_top_fix)
    opt_opt <- ncol(matrix_top_opt)
  }


  return(c(matrix_top_fix[,opt_fix], matrix_top_opt[,opt_opt]))

}
```

`ProgPredLasso` *Identification of prognostic and predictive biomarkers*

Description

The computes the regularization path of the Prognostic Predictive Lasso described in the paper Zhu et al. (2022) given in the references.

Usage

```
ProgPredLasso(X1, X2, Y=Y, cor_matrix=NULL, gamma=0.99, maxsteps=500)
```

Arguments

`X1`	Design matrix of patients characteristics with treatment 1
`X2`	Design matrix of patients characteristics with treatment 2
`Y`	Response variable
`cor_matrix`	Correlation matrix of biomarkers. If not specified, the function `cvCovEst` from package `cvCovEst` will be used to estimate this matrix.
`gamma`	Parameter γ defined in the paper Zhu et al. (2020) given in the references. Its default value is 0.99.
`maxsteps`	Integer specifying the maximum number of steps for the generalized Lasso algorithm. Its default value is 500.

Value

Returns a list with the following components

`lambda`	different values of the parameter λ considered.
`beta`	matrix of the estimations of β for all the λ considered.
`beta.min`	estimation of β which minimize the MSE.
`bic`	BIC for all the λ considered.
`mse`	MSE for all the λ considered.

Author(s)

Wencan Zhu, Celine Levy-Leduc, Nils Ternes

Examples

```
X1 = t(sapply(c(1:25),FUN=function(x) rnorm(50)))
X2 = t(sapply(c(1:25),FUN=function(x) rnorm(50)))
Y=rnorm(50)
ProgPredLasso(X1=X1, X2=X2, Y=Y)

## The function is currently defined as
function(X1, X2, Y=Y, cor_matrix=NULL, gamma=0.99, maxsteps=500){
  p10=ncol(X1)
  p20=ncol(X2)
  p2 <- p10+p20

  mat_coef <- matrix(0,p2, p2)
  mat_coef[1:p10, 1:p10] <- diag(rep(1,p10))
  mat_coef[(p10+1):(p2), 1:p10] <- diag(rep(-1,p10))
  mat_coef[(p10+1):(p2), (p10+1):(p2)] <- diag(rep(1,p10))

  n1=nrow(X1)
  n2=nrow(X2)
  n=n1+n2

  if((n1+n2)!=length(Y))
    stop("the sample size should be consistent")

  TRT1  <- c(rep(1,n1), rep(0, n2))
  TRT2  <- c(rep(0,n1), rep(1, n2))

  X_full <- cbind(rbind(X1, X2)*TRT1, rbind(X1, X2)*TRT2)
  X_classic <- cbind(rbind(X1, X2), rbind(X1, X2)*TRT2)

  if(is.null(cor_matrix)){
    X_all <- rbind(X1, X2)
    cv_cov_est_out <- cvCovEst(
      dat = X_all,
      estimators = c(
        linearShrinkLWEst, denseLinearShrinkEst,
        thresholdingEst, poetEst, sampleCovEst
```

```
    ),
    estimator_params = list(
      thresholdingEst = list(gamma = c(0.2, 0.4)),
      poetEst = list(lambda = c(0.1, 0.2), k = c(1L, 2L))
    ),
    cv_loss = cvMatrixFrobeniusLoss,
    cv_scheme = "v_fold",
    v_folds = 5
  )
  cor_matrix <- cov2cor(cv_cov_est_out$estimate)
}

cor_matrix_full = matrix(0, (p2), (p2))
cor_matrix_full[1:p10, 1:p10] = cor_matrix_full[((p10+1):(p2)), ((p10+1):(p2))] = cor_matrix

cor_matrix_full <- round(cor_matrix_full, 6)
file <- try(SVD_sigma <- svd(cor_matrix_full))
if (class(file) == "try-error") {
  cat("Caught an error during SVD.\n")
  Eigen_Sigma <- eigen(cor_matrix_full)
  V_sigma <- Eigen_Sigma$vectors
  lam <- Eigen_Sigma$values
  square_root_sigma <- V_sigma
  inv_diag <- ifelse(lam<0.000001, 0, 1/sqrt(lam))
  inv_square_root_Sigma <- V_sigma
} else {
  U_sigma <- SVD_sigma$u
  D_sigma <- SVD_sigma$d
  square_root_sigma <- U_sigma
  inv_diag <- ifelse(D_sigma<0.000001, 0, 1/sqrt(D_sigma))
  inv_square_root_Sigma <- U_sigma
  inv_square_root_Sigma <- U_sigma
}

#tranformation matrix
inv_square_root_Sigma_trt = diag(1, ncol=(p2+2), nrow=(p2+2))
inv_square_root_Sigma_trt[3:(p2+2), 3:(p2+2)] <- inv_square_root_Sigma

mat_trt <- matrix(0,nrow=p2, ncol=(p2+2) )
mat_trt[, 3:(p2+2)] <- mat_coef
mat_trt[, c(1,2)] <- matrix(0, p2, 2)

X_new <- cbind(c(rep(1, n1),rep(0, n2)), c(rep(0, n1),rep(1, n2)), X_full)
X_new2 <- cbind(c(rep(1, n1),rep(0, n2)), c(rep(0, n1),rep(1, n2)), X_classic)
trans_mat <- mat_trt%*%inv_square_root_Sigma_trt
X_tilde0 <- X_new%*%inv_square_root_Sigma_trt
out0 <-  genlasso(Y, X_tilde0, trans_mat, maxsteps=maxsteps)

if(p10<=50){
  top_grill <- seq(1, p10, 2)
} else if(p10<210){
  top_grill <- c(1:50, seq(52,p10, 2))
} else if (p10<500){
```

```
  top_grill <- c(1:50, seq(52,100, 2), seq(105,200, 5), seq(210, p10, 10))
 } else {
  top_grill <- c(1:50, seq(52,100, 2), seq(105,200, 5), seq(210, 500, 10))
 }

 opt_top <- opt_final_top <- c()
 beta_final <- matrix(NA, length(out0$lambda), (p2+2))
 mse_final = bic_final = c()
 for(i in 1:length(out0$lambda)){
  #### Filter on beta_tilde
  gamma_tilde <- out0$beta[,i][-c(1,2)]
  gamma_tilde_prog <- gamma_tilde[1:p10]
  gamma_tilde_pred <- gamma_tilde[(p10+1):(p2)]

  gamma_tilde_opt <-Correction2Vect(X=X_tilde0, Y=Y, te=out0$beta[c(1,2),i],
  vector_prog =gamma_tilde_prog, vector_pred= gamma_tilde_pred ,delta = 0.99999,
  top_grill. = top_grill)

  #### Filter on beta
  beta_final0 <- mat_coef%*%inv_square_root_Sigma%*%gamma_tilde_opt
  beta_prog <- beta_final0[1:p10]
  beta_pred <- beta_final0[(p10+1):(p2)]

  beta_opt_final <- Correction2Vect(X=X_new2, Y=Y, te=out0$beta[c(1,2),i],
  vector_prog =beta_prog, vector_pred= beta_pred ,delta = 0.99999,
  top_grill. = top_grill, toZero = TRUE)

  beta_final[i, ] <- round(c(out0$beta[c(1,2),i], beta_opt_final),6)

  X_temp <- X_classic[, which(beta_opt_final!=0)]
  if(length(which(beta_opt_final!=0))>=(length(Y)-2)){
  X_pred <- as.matrix(cbind(c(rep(1, n1),rep(0, n2)), c(rep(0, n1),rep(1, n2)), X_temp))
    mod_ridge <- cv.glmnet(x=as.matrix(X_pred), y=Y, alpha=0, intercept=FALSE)
    mse_final[i] <- min(mod_ridge$cvm)
    bic_final[i] <- n*log(mse_final[i])+log(n)*length(which(beta_opt_final!=0))
  } else {
  mydata <- data.frame(Y=Y, cbind(c(rep(1, n1),rep(0, n2)), c(rep(0, n1),rep(1, n2)), X_temp))
   formula <- paste0("Y~-1 +",paste0(colnames(mydata[, -which(colnames(mydata)=="Y")]),
    collapse=" + "))
    myform <- as.formula(formula)
    mod_lm <- lm(myform, data=mydata)
    mse_final[i] <- mean(mod_lm$residuals^2)
    bic_final[i] <- n*log(mse_final[i])+log(n)*length(which(beta_opt_final!=0))
  }
 }

 beta_min <- beta_final[which.min(bic_final), ]
 return(list(lambda=out0$lambda, beta=beta_final, beta.min=beta_min, bic=bic_final, mse=mse_final))
}
```

---

top *Thresholding to 0*

Description

This function keeps only the K largest values of the vector and sets the others to 0.

Usage

```
top(vect, thresh)
```

Arguments

vect	vector to threshold
thresh	threshold

Value

This function returns the thresholded vector.

Author(s)

Wencan Zhu, Celine Levy-Leduc, Nils Ternes

Examples

```
x=sample(1:10,10)
thresh=3
top(x,thresh)

## The function is currently defined as
function(vect, thresh){
  sorted_vect <- sort(abs(vect),decreasing=TRUE)
  v<-sorted_vect[thresh]
  ifelse(abs(vect)>=v,vect,0)
}
```

---

top_thresh *Thresholding to a given threshold of the smallest values*

Description

This function keeps only the K largest values of the vector and sets the others to the smallest value among the K largest.

Usage

```
top_thresh(vect,thresh)
```

Arguments

vect	vector to threshold
thresh	threshold

Value

This function returns the thresholded vector.

Author(s)

Wencan Zhu, Celine Levy-Leduc, Nils Ternes

Examples

```
x=sample(1:10,10)
thresh=3
top_thresh(x,thresh)

## The function is currently defined as
function (vect, thresh)
{
    sorted_vect <- sort(abs(vect),decreasing=TRUE)
    v = sorted_vect[thresh]
    ifelse(abs(vect) >= v, vect, v)
  }
```

Index