```
library("glmnet")
library("survival")
library("mvtnorm")
library("glasso")
library("lpSolve")

coxhdi <- function(x,time,status,method1="W",cptw="L",indx = c(1))
{

#########################################################################################################

  ##Input values:
##
  ##x:       Covariates
##
  ##time:    Survival times or censoring times
##
  ##status:  Censoring status
  ##method1: "S","W" or "L", represents Score, Wald or Likelihood Ratio Test
##
  ##cptw:    "L" or "D", represents using either Lasso or Dantzig method to compute w (decorrelation
vector),##
  ##         where Lasso gives faster computation
##
  ##indx:    Set of indices to conduct testing

#########################################################################################################


  Pval = c()
  d    = dim(x)[2]
  n    = dim(x)[1]
  for (coi in indx)
  {
    lambdas = seq(from = 0.1,to=1.5,by = 0.03)
    lambdas = lambdas*log(d)/n                 ##Set of tuning parameters
    cv.fit <- cv.glmnet(x, Surv(time, status),family = "cox", lambda = lambdas, maxit = 10000,nfolds =
5,alpha=1)
    fit     = cv.fit$glmnet.fit
    tmp     = which(fit$lambda == cv.fit$lambda.min)

    if (sum(fit$beta[,tmp]!=0)==0)
    {
      beta = rep(0,d)
    } else {
      beta    = fit$beta[,tmp]   # Initial Estimator
    }
    betas   = beta
    betas[coi] = 0              # H0: beta_i = 0

    stime = sort(time)          # Sorted survival/censored times
    otime = order(time)         # Order of time

    Vs  = matrix(rep(0,d*d),nrow = d)
    Hs  = Vs                                # Hessian
    ind = 0

    la  = rep(0,n)                          # Gradient w.r.t parameter of interest
    lb  = matrix(rep(0,(d-1)*n),nrow = n)    # Gradient w.r.t nuisance parameter (theta)
    i   = 1
    while( i<=n)
    {
      if (status[otime[i]]==1)
      {
        ind = which(time >= stime[i])
        S0  = 0
        S1  = rep(0,d)
        S2  = matrix(rep(0,d*d),nrow = d)

        if (length(ind)>0)
        {
          for (j in 1:length(ind))
          {
            tmp = exp(x[ind[j],]%*%betas)
```

```
        S0  = S0 + tmp

        S1  = S1 + tmp %*%t(x[ind[j],])

        tmp = apply(tmp,1,as.numeric)
        S2  = S2 + tmp*x[ind[j],]%*%t(x[ind[j],])
      }
    }
    S0 = apply(S0,1,as.numeric)

    la[i]  = -(x[otime[i],coi] - S1[coi]/S0)
    if (coi == 1)
    {
      lb[i,] = -(x[otime[i],c((coi+1):d)] - S1[c((coi+1):d)]/S0)
    } else if (coi == d){
      lb[i,] = -(x[otime[i],c(1:(coi-1))] - S1[c(1:(coi-1))]/S0)
    } else {
      lb[i,] = -(x[otime[i],c(1:(coi-1), (coi+1):d)] - S1[c(1:(coi-1), (coi+1):d)]/S0)
    }
    V   = S0*S2 - t(S1)%*%(S1)
    Hs  = Hs + V/(n*S0^2)
  }
  i = i + 1
}

#Hs = Hs/n

if (cptw == "L")
{
  if (method1=="L")
  {
    cv.fit = glmnet(lb,la,standardize = FALSE,intercept = FALSE)
    fit     = cv.fit$glmnet.fit
    tmp     = which(fit$lambda == cv.fit$lambda.min)
    if (sum(fit$beta[,tmp]!=0)==0)
    {
      what = rep(0,d-1)
    } else {
      what    = fit$beta[,tmp]
    }
  } else {
    fit    = glmnet(lb,la,standardize = FALSE,intercept = FALSE)
    lgth = length(fit$lambda)
    HBIC = rep(10000,lgth)
    for (tmp in 1:lgth)
    {
      if (sum(fit$beta[,tmp])!=0)
      {
        loss = sum((la-lb%*%fit$beta[,tmp])^2)/n
        HBIC[tmp] = log(loss) + fit$df[tmp]*log(d)/n*eta*(log(log(n)))
      }
    }
    tmp = which(HBIC==min(HBIC))
    what = fit$beta[,tmp]
  }


} else if (cptw == "D") {
  if (coi == 1)
  {
    Hab = Hs[(coi+1):d,coi]
    Hbb = Hs[(coi+1):d,(coi+1):d]
  } else if (coi == d){
    Hab = Hs[1:(coi-1),coi]
    Hbb = Hs[1:(coi-1),1:(coi-1)]
  } else{
    Hab = Hs[c(1:(coi-1),(coi+1):d),coi]
    Hbb = Hs[c(1:(coi-1),(coi+1):d),c(1:(coi-1),(coi+1):d)]
  }

  tmp = matrix(rep(0,(d-1)*(d-1)),nrow = d-1)
  A1  = cbind(Hbb,tmp)
  A2  = cbind(-Hbb,tmp)
  A3  = cbind(diag(d-1),-diag(d-1))
```

```
      A4  = cbind(-diag(d-1),-diag(d-1))

      A   = rbind(A1,A2,A3,A4)
      obj = c(rep(0,d-1),rep(1,d-1))
      dir = rep("<=",(d-1)*2 + (d-1)*2)
      lambda = 1*sqrt(log(d)/n)
      rhs = c(Hab + lambda, -Hab + lambda, rep(0,d-1),rep(0,d-1))
      tmp = lp(direction = "min",objective.in = obj,const.mat = A,const.dir = dir, const.rhs = rhs)
      what = tmp$solution[1:(d-1)]
    }

    what = matrix(what,nrow=d-1)
    if (method1 == "S"){
      # Decorrelated Score
      if (coi == 1)
      {
        var   = max(Hs[coi,coi] - t(what)%*%Hs[c((coi+1):d),coi],0.1)
      } else if (coi == d){
        var   = max(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-1)),coi],0.1)
      } else {
      var   = max(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-1),(coi+1):d),coi],0.1)
      }
      lbt = colSums(lb)
      lbt = matrix(lbt,nrow = d-1)
      S = sum(la) - t(what)%*%(lbt)
      S = S/n

      tmp   = sqrt(n)*S/sqrt(var)
      pval  = 2*pnorm(-abs(tmp))
    } else if (method1 == "W") {
      # Decorrelated Wald
      if (coi == 1)
      {
        S = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] -
t(what)%*%Hs[c((coi+1):d),coi])
        var   = Hs[coi,coi] - t(what)%*%Hs[c((coi+1):d),coi]
      } else if (coi == d){
        S = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-
1)),coi])
        var   = Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-1)),coi]
      } else {
        S = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-1),
(coi+1):d),coi])
        var   = Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-1),(coi+1):d),coi]
      }
      tmp   = (n)*S^2*(max(var,1e-8))
      pval  = 1-pchisq(tmp,1)
    } else if (method1 == "L") {
      # Deccorelated LHR
      tmp = which(la!=0)


      if (coi == 1)
      {
        atilde = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] -
t(what)%*%Hs[c((coi+1):d),coi])
        beta1 = c(atilde,beta[(coi+1):d]-atilde*what[(coi):(d-1)])
      } else if (coi == d) {
        atilde = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-
1)),coi])
        beta1 = c(beta[1:(coi-1)]-atilde*what[1:(coi-1)],atilde)
      } else {
        atilde = beta[coi] - (mean(la)  - t(what)%*%(colMeans(lb)))/(Hs[coi,coi] - t(what)%*%Hs[c(1:(coi-
1),(coi+1):d),coi])
        beta1 = c(beta[1:(coi-1)]-atilde*what[1:(coi-1)],atilde,beta[(coi+1):d]-atilde*what[(coi):(d-1)])
      }


      lossa  = 0    #Log-Likelihood under alternative
      i = 1
      while( i<=n)
      {
        ind = which(time >= stime[i])
        S0  = 0
```

```
        # S1  = matrix(rep(0,d),nrow = d)
        if (length(ind)>0)
        {
          for (j in 1:length(ind))
          {
            tmp = 1/n * exp(x[ind[j],]%*%beta1)
            tmp = apply(tmp,1,as.numeric)
            S0  = S0 + tmp
          }
        }
        if (status[otime[i]]==1)
        {
          lossa = lossa + 1/n*(x[otime[i],]%*%beta1 - log(S0))
        }
        i = i + 1
      }
      lossn  = 0  #Log-Likelihood under null
      i = 1
      if (coi == 1)
      {
        beta2 = c(0,beta[(coi+1):d])
      } else if (coi == d) {
        beta2 = c(beta[1:(coi-1)],0)
      } else {
      beta2 = c(beta[1:(coi-1)],0,beta[(coi+1):d])
      }
      while( i<=n)
      {
        ind = which(time >= stime[i])
        S0  = 0
        if (length(ind)>0)
        {
          for (j in 1:length(ind))
          {
            tmp = 1/n * exp(x[ind[j],]%*%beta2)
            tmp = apply(tmp,1,as.numeric)
            S0  = S0 + tmp
          }
        }

        if (status[otime[i]]==1)
        {
          lossn = lossn + 1/n*(x[otime[i],]%*%beta2 - log(S0))
        }
        i = i + 1
      }
      tmp   = n*lossa - n*lossn
      tmp   = 2*tmp
      pval  = 1-pchisq(tmp,1)
    }
    Pval = c(Pval,pval)
  }
  return(Pval)
}
```