CRC-DATA-WEIBULL

```
##########Application of project 2###############
#rm(list=ls())
#for Bernstein Polynomials:
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(survival)
require(stats)
library(splines2)
library(pracma)  ## for numerical differentiation
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
# setwd("/Users/fatemehmahmoudi/Desktop/Codes-bin")

# source("sim-functions-weibull.R")
# source("varsel-functions-weibull.R")

logLike.weibull.SCR.SM.LT <- function(para, y1, y2, delta1, delta2, l, Xmat1=NULL, Xmat2=NULL, Xmat3=NU
{
  ##
  kappa1    <- exp(para[1])
  alpha1 <- exp(para[2])
  kappa2    <- exp(para[3])
  alpha2 <- exp(para[4])
```

```
kappa3    <- exp(para[5])
alpha3 <- exp(para[6])
if(frailty == TRUE){
  theta    <- exp(para[7])
  thetaInv <- 1 / theta
}
##
nP.0 <- ifelse(frailty, 7, 6)
nP.1 <- ncol(Xmat1)
nP.2 <- ncol(Xmat2)
nP.3 <- ncol(Xmat3)
##
eta.1 <- as.vector(Xmat1 %*% para[nP.0 + c(1:nP.1)])
eta.2 <- as.vector(Xmat2 %*% para[nP.0 + nP.1 + c(1:nP.2)])
eta.3 <- as.vector(Xmat3 %*% para[nP.0 + nP.1 + nP.2 + c(1:nP.3)])
##
type1 <- as.numeric(delta1 == 1 & delta2 == 1 & l < y1)
type2 <- as.numeric(delta1 == 0 & delta2 == 1 & l < y1)
type3 <- as.numeric(delta1 == 1 & delta2 == 0 & l < y1)
type4 <- as.numeric(delta1 == 0 & delta2 == 0 & l < y1)
type5 <- as.numeric(delta1 == 1 & delta2 == 1 & y1 <= l & l < y2)
type6 <- as.numeric(delta1 == 1 & delta2 == 0 & y1 <= l & l < y2)
##
log.h1star.y1 <- log(alpha1) + log(kappa1) + (alpha1 - 1) * log(y1) + eta.1
log.h2star.y1 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y1) + eta.2
log.h2star.y2 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y2) + eta.2
log.h3star.y2 <- log(alpha3) + log(kappa3) + (alpha3 - 1) * log(y2-y1) + eta.3
##
q.y1 <- kappa1*(y1)^alpha1 * exp(eta.1) + kappa2*(y1)^alpha2 * exp(eta.2)
q.y2 <- kappa1*(y2)^alpha1 * exp(eta.1) + kappa2*(y2)^alpha2 * exp(eta.2)
q.l <- kappa1*(l)^alpha1 * exp(eta.1) + kappa2*(l)^alpha2 * exp(eta.2)
##
w.y1.y2 <- kappa3*(y2-y1)^alpha3 * exp(eta.3)
w.y1.l  <- kappa3*((l-y1)^(alpha3))* exp(eta.3)
##
k1 <- w.y1.y2
k2.y1 <- q.y1 - q.l
k2.y2 <- q.y2 - q.l
k3 <- w.y1.y2 - w.y1.l
##
if(frailty == TRUE)
{
  logLike1 <- log.h1star.y1 + log.h3star.y2 + log(1+theta) - ((thetaInv + 2) * log(1 + (theta * (k1 +
  logLike2 <- log.h2star.y1 - ((thetaInv + 1) * log(1 + (theta * k2.y1)))  ## Making in terms of y1
  logLike3 <- log.h1star.y1 - ((thetaInv + 1) * log(1 + (theta * (k1 + k2.y1))))
  logLike4 <- - thetaInv * log(1 + (theta * k2.y1))  ## Making in terms of y1
  logLike5 <- log.h3star.y2 - ((thetaInv + 1) * log(1 + (theta * k3)))
  logLike6 <- - thetaInv * log(1 + (theta * k3))
}
if(frailty == FALSE)
{
  logLike1 <- log.h1star.y1 + log.h3star.y2 - (k1 + k2.y1)
  logLike2 <- log.h2star.y1 - k2.y1  ## Making in terms of y1
```

```
    logLike3 <- log.h1star.y1 - (k1 + k2.y1)
    logLike4 <- - k2.y1  ## Making in terms of y1
    logLike5 <- log.h3star.y2 - k3
    logLike6 <- - k3
  }
  ##
  loglh <- sum(logLike1[type1==1]) + sum(logLike2[type2==1]) + sum(logLike3[type3==1]) + sum(logLike4[ty
  ##
  return(-loglh)
}
dlogLike.weibull.new <- function(para, y1, y2, delta1, delta2, l, Xmat1=NULL, Xmat2=NULL, Xmat3=NULL, fi
{
  ##
  kappa1 <- exp(para[1])
  alpha1 <- exp(para[2])
  kappa2 <- exp(para[3])
  alpha2 <- exp(para[4])
  kappa3 <- exp(para[5])
  alpha3 <- exp(para[6])
  if(frailty == TRUE){
    theta    <- exp(para[7])
    thetaInv <- 1 / theta
  }
  ##
  nP.0 <- ifelse(frailty, 7, 6)
  nP.1 <- ncol(Xmat1)
  nP.2 <- ncol(Xmat2)
  nP.3 <- ncol(Xmat3)
  ##
  eta.1 <- as.vector(Xmat1 %*% para[nP.0 + c(1:nP.1)])
  eta.2 <- as.vector(Xmat2 %*% para[nP.0 + nP.1 + c(1:nP.2)])
  eta.3 <- as.vector(Xmat3 %*% para[nP.0 + nP.1 + nP.2 + c(1:nP.3)])
  ##
  type1 <- as.numeric(delta1 == 1 & delta2 == 1 & l < y1)
  type2 <- as.numeric(delta1 == 0 & delta2 == 1 & l < y1)
  type3 <- as.numeric(delta1 == 1 & delta2 == 0 & l < y1)
  type4 <- as.numeric(delta1 == 0 & delta2 == 0 & l < y1)
  type5 <- as.numeric(delta1 == 1 & delta2 == 1 & y1 <= l & l < y2)
  type6 <- as.numeric(delta1 == 1 & delta2 == 0 & y1 <= l & l < y2)
  ##
  log.h1star.y1 <- log(alpha1) + log(kappa1) + (alpha1 - 1) * log(y1) + eta.1
  log.h2star.y1 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y1) + eta.2
  log.h2star.y2 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y2) + eta.2
  log.h3star.y2 <- log(alpha3) + log(kappa3) + (alpha3 - 1) * log(y2-y1) + eta.3
  ##
  q.y1 <- kappa1*(y1)^alpha1 * exp(eta.1) + kappa2*(y1)^alpha2 * exp(eta.2)
  q.y2 <- kappa1*(y2)^alpha1 * exp(eta.1) + kappa2*(y2)^alpha2 * exp(eta.2)
  q.l <- kappa1*(l)^alpha1 * exp(eta.1) + kappa2*(l)^alpha2 * exp(eta.2)
  ##
  w.y1.y2 <- kappa3*(y2-y1)^alpha3 * exp(eta.3)
  w.y1.l  <- kappa3*(l-y1)^alpha3 * exp(eta.3)
  ##
  k1 <- w.y1.y2
```

```
k2.y1 <- q.y1 - q.l
k2.y2 <- q.y2 - q.l
k3 <- w.y1.y2 - w.y1.l
##
if(frailty == TRUE)
{

  n=NROW(eta.1)
  p=ncol(Xmat1)
  #score_ij-->i(l_i),(j:beta_j)--->score23:dl_2(\phi)/d\beta_3.
  #we have l_i; i=1,2,3,4, and beta_j; j=1,2,3,4.
  A1=(kappa1*(y1)^alpha1*exp(eta.1))-(kappa1*(l)^alpha1*exp(eta.1))
  denom1=1+(theta*(k1+k2.y1))
  denom2=1+(theta*k2.y1)
  A2=(kappa2*(y1)^alpha2 *exp(eta.2))-(kappa2*(l)^alpha2 *exp(eta.2))
  A3=(kappa3*(y2-y1)^alpha3 *exp(eta.3))


  #each of these below is a n*1 column vector.
  aa1=1-((1+2*theta)*A1/denom1)
  aa2=-(1+theta)*(A1)/(denom2)
  aa3=1-((1+theta)*(A1)/(denom1))
  aa4=-(A1)/(denom2)
  bb1=-(1+2*theta)*(A2)/(denom1)
  bb2=1-(((1+theta)*A2)/(denom2))
  bb3=-(1+theta)*(A2)/(denom1)
  bb4=-(A2)/(denom2)
  cc1=1-(((1+2*theta)*A3)/(denom1))
  cc3=-(1+theta)*(A3)/(denom1)


  #now, we multiply a X_ik; k=1,2,3, i=1,2,...,n in those aa1, aa2, ... to construct the components o
  #X_ik is a p-vector in the column form.
  #So, by multiplying, we have a matrix of n*p for each of score_k; k=1,2,3.
  #each score_k; k=1,2,3 consists of 4 parts that are the components in the likelihood function.
  #4 components of score1:
  score11=matrix(NA,n,p)
  score12=matrix(NA,n,p)
  score13=matrix(NA,n,p)
  score14=matrix(NA,n,p)
  #4 components of score2:
  score21=matrix(NA,n,p)
  score22=matrix(NA,n,p)
  score23=matrix(NA,n,p)
  score24=matrix(NA,n,p)
  #2 components of score3 (2 of them are zero):
  score31=matrix(NA,n,p)
  score33=matrix(NA,n,p)

  for (i in 1:n){
    #each row in the n*p matrix is the multipication of each element ofaa1, bb1, or etc for i=1,2,...
    score11[i,]=as.vector(aa1[i]*Xmat1[i,])
    score12[i,]=as.vector(aa2[i]*Xmat1[i,])
```

```
      score13[i,]=as.vector(aa3[i]*Xmat1[i,])
      score14[i,]=as.vector(aa4[i]*Xmat1[i,])

      score21[i,]=as.vector(bb1[i]*Xmat2[i,])
      score22[i,]=as.vector(bb2[i]*Xmat2[i,])
      score23[i,]=as.vector(bb3[i]*Xmat2[i,])
      score24[i,]=as.vector(bb4[i]*Xmat2[i,])

      score31[i,]=as.vector(cc1[i]*Xmat3[i,])
      score33[i,]=as.vector(cc3[i]*Xmat3[i,])
    }
    #each of the score1, score2, and score3 below are of .
    #Now, by score_k; k=1,2,3, we have 3 matrices each of n*p dimension:
    score1=(score11*(type1==1))+(score12*(type2==1))+(score13*(type3==1))+(score14*(type4==1))
    score2=(score21*(type1==1))+(score22*(type2==1))+(score23*(type3==1))+(score24*(type4==1))
    score3=(score31*(type1==1))+(score33*(type3==1))

    #now we sum over i=1,2,...,n which means that we sum over each column so that we have 1*p vector fo
    score1.final=colSums(score1)
    score2.final=colSums(score2)
    score3.final=colSums(score3)
    #finally, score is a vecrtor consisiting of 3 p-vectors:
    score=c(score1.final,score2.final,score3.final)

  }


  return(-score)
}


ddlogLike.weibull.new <- function(para, y1, y2, delta1, delta2, l, Xmat1=NULL, Xmat2=NULL, Xmat3=NULL, 
{
  ##
  kappa1 <- exp(para[1])
  alpha1 <- exp(para[2])
  kappa2 <- exp(para[3])
  alpha2 <- exp(para[4])
  kappa3 <- exp(para[5])
  alpha3 <- exp(para[6])
  if(frailty == TRUE){
    theta    <- exp(para[7])
    thetaInv <- 1 / theta
  }
  ##
  nP.0 <- ifelse(frailty, 7, 6)
  nP.1 <- ncol(Xmat1)
  nP.2 <- ncol(Xmat2)
  nP.3 <- ncol(Xmat3)
  ##
  eta.1 <- as.vector(Xmat1 %*% para[nP.0 + c(1:nP.1)])
  eta.2 <- as.vector(Xmat2 %*% para[nP.0 + nP.1 + c(1:nP.2)])
  eta.3 <- as.vector(Xmat3 %*% para[nP.0 + nP.1 + nP.2 + c(1:nP.3)])
```

```
##
type1 <- as.numeric(delta1 == 1 & delta2 == 1 & l < y1)
type2 <- as.numeric(delta1 == 0 & delta2 == 1 & l < y1)
type3 <- as.numeric(delta1 == 1 & delta2 == 0 & l < y1)
type4 <- as.numeric(delta1 == 0 & delta2 == 0 & l < y1)
type5 <- as.numeric(delta1 == 1 & delta2 == 1 & y1 <= l & l < y2)
type6 <- as.numeric(delta1 == 1 & delta2 == 0 & y1 <= l & l < y2)
##
log.h1star.y1 <- log(alpha1) + log(kappa1) + (alpha1 - 1) * log(y1) + eta.1
log.h2star.y1 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y1) + eta.2
log.h2star.y2 <- log(alpha2) + log(kappa2) + (alpha2 - 1) * log(y2) + eta.2
log.h3star.y2 <- log(alpha3) + log(kappa3) + (alpha3 - 1) * log(y2-y1) + eta.3
##
q.y1 <- kappa1*(y1)^alpha1 * exp(eta.1) + kappa2*(y1)^alpha2 * exp(eta.2)
q.y2 <- kappa1*(y2)^alpha1 * exp(eta.1) + kappa2*(y2)^alpha2 * exp(eta.2)
q.l <- kappa1*(l)^alpha1 * exp(eta.1) + kappa2*(l)^alpha2 * exp(eta.2)
##
w.y1.y2 <- kappa3*(y2-y1)^alpha3 * exp(eta.3)
w.y1.l  <- kappa3*(l-y1)^alpha3 * exp(eta.3)
##
k1 <- w.y1.y2
k2.y1 <- q.y1 - q.l
k2.y2 <- q.y2 - q.l
k3 <- w.y1.y2 - w.y1.l


if(frailty == TRUE)
{
  n=NROW(eta.1)
  p=ncol(Xmat1)
  #score_ij-->i(l_i),(j:beta_j)--->score23:dl_2(\phi)/d\beta_3.
  #we have l_i; i=1,2,3,4, and beta_j; j=1,2,3,4.
  A1=(theta*kappa1*(y1)^alpha1*exp(eta.1))-(theta*kappa1*(l)^alpha1*exp(eta.1))
  A2=(theta*kappa2*(y1)^alpha2 *exp(eta.2))-(theta*kappa2*(l)^alpha2 *exp(eta.2))
  A3=theta*(kappa3*(y2-y1)^alpha3 *exp(eta.3))
  A4=kappa1*(y1^alpha1-l^alpha1)*exp(eta.1)
  A5=(kappa1*(y1)^alpha1*exp(eta.1))-(kappa1*(l)^alpha1*exp(eta.1))
  A6=(kappa2*(y1)^alpha2 *exp(eta.2))-(kappa2*(l)^alpha2 *exp(eta.2))
  A7=(kappa3*(y2-y1)^alpha3 *exp(eta.3))

  denom1=1+(theta*(k1+k2.y1))
  denom2=1+(theta*k2.y1)

  #All n*1 column vectors:
  #dbeta1beta1
  B111=-(1+2*theta)*A4*((denom1-A1)/(denom1^2))
  B211=-(theta+1)*A4*((denom2-A1)/(denom2^2))
  B311=-(1+theta)*A4*((denom1-A1)/(denom1^2))
  B411=-A4*((denom2-A1)/(denom2^2))
  #dbeta1beta2:
  B112=(2*theta+1)*((A2*A5)/(denom1^2))
  B212=(theta+1)*((A2*A5)/(denom2^2))
  B312=(theta+1)*((A2*A5)/(denom1^2))
```

```
B412=(A2*A5)/(denom2^2)
#dbeta1beta3:
B113=(2*theta+1)*((A3*A4)/(denom1^2))
B213=0
B313=(1+theta)*((A3*A4)/(denom1^2))
B413=0
#dbeta2beta3:
B123=(1+2*theta)*((A3*A6)/(denom1^2))
B223=0
B323=(1+theta)*((A3*A6)/(denom1^2))
B423=0
#dbeta2beta2
B122=-(1+2*theta)*((A6*(denom1-A2))/denom1^2)
B222=-(1+theta)*((A6*(denom2-A2))/denom2^2)
B322=-(1+theta)*((A6*(denom1-A2))/denom1^2)
B422=-((A6*(denom2-A2))/denom2^2)
#dbeta3beta3:
B133=-(1+2*theta)*((A7*(denom1-A3))/(denom1^2))
B333=-(1+theta)*((A7*(denom1-A3))/(denom1^2))


dscore111=matrix(0,p,p)
dscore211=matrix(0,p,p)
dscore311=matrix(0,p,p)
dscore411=matrix(0,p,p)
dscore112=matrix(0,p,p)
dscore212=matrix(0,p,p)
dscore312=matrix(0,p,p)
dscore412=matrix(0,p,p)
dscore113=matrix(0,p,p)
dscore313=matrix(0,p,p)
dscore123=matrix(0,p,p)
dscore323=matrix(0,p,p)
dscore122=matrix(0,p,p)
dscore222=matrix(0,p,p)
dscore322=matrix(0,p,p)
dscore422=matrix(0,p,p)
dscore133=matrix(0,p,p)
dscore333=matrix(0,p,p)

for (i in 1:n){
  #all p*p matrices:
  #dbeta1dbeta1:
  dscore111=dscore111+(B111[i]*(Xmat1[i,]%*%t(Xmat1[i,])))*((type1==1)[i])
  dscore211=dscore211+(B211[i]*(Xmat1[i,]%*%t(Xmat1[i,])))*((type2==1)[i])
  dscore311=dscore311+(B311[i]*(Xmat1[i,]%*%t(Xmat1[i,])))*((type3==1)[i])
  dscore411=dscore411+(B411[i]*(Xmat1[i,]%*%t(Xmat1[i,])))*((type4==1)[i])
  #dbeta1dbeta2:
  dscore112=dscore112+(B112[i]*(Xmat1[i,]%*%t(Xmat2[i,])))*((type1==1)[i])
  dscore212=dscore212+(B212[i]*(Xmat1[i,]%*%t(Xmat2[i,])))*((type2==1)[i])
  dscore312=dscore312+(B312[i]*(Xmat1[i,]%*%t(Xmat2[i,])))*((type3==1)[i])
  dscore412=dscore412+(B412[i]*(Xmat1[i,]%*%t(Xmat2[i,])))*((type4==1)[i])
  #dbeta1dbeta3:
```

```
      dscore113=dscore113+(B113[i]*(Xmat1[i,]%*%t(Xmat3[i,])))*((type1==1)[i])
      # dscore213=dscore213+(B213[i]*(Xmat1[i,]%*%t(Xmat3[i,])))*((type2==1)[i])
      dscore313=dscore313+(B313[i]*(Xmat1[i,]%*%t(Xmat3[i,])))*((type3==1)[i])
      # dscore413=dscore413+(B413[i]*(Xmat1[i,]%*%t(Xmat3[i,])))*((type4==1)[i])
      #dbeta2dbeta3:
      dscore123=dscore123+(B123[i]*(Xmat2[i,]%*%t(Xmat3[i,])))*((type1==1)[i])
      # dscore223=dscore223+(B223[i]*(Xmat2[i,]%*%t(Xmat3[i,])))*((type2==1)[i])
      dscore323=dscore323+(B323[i]*(Xmat2[i,]%*%t(Xmat3[i,])))*((type3==1)[i])
      # dscore423=dscore423+(B423[i]*(Xmat2[i,]%*%t(Xmat3[i,])))*((type4==1)[i])
      #dbeta2dbeta2:
      dscore122=dscore122+(B122[i]*(Xmat2[i,]%*%t(Xmat2[i,])))*((type1==1)[i])
      dscore222=dscore222+(B222[i]*(Xmat2[i,]%*%t(Xmat2[i,])))*((type2==1)[i])
      dscore322=dscore322+(B322[i]*(Xmat2[i,]%*%t(Xmat2[i,])))*((type3==1)[i])
      dscore422=dscore422+(B422[i]*(Xmat2[i,]%*%t(Xmat2[i,])))*((type4==1)[i])
      #dbeta3dbeta3:
      dscore133=dscore133+(B133[i]*(Xmat3[i,]%*%t(Xmat3[i,])))*((type1==1)[i])
      # dscore233=dscore233+(B233[i]*(Xmat3[i,]%*%t(Xmat3[i,])))*((type2==1)[i])
      dscore333=dscore333+(B333[i]*(Xmat3[i,]%*%t(Xmat3[i,])))*((type3==1)[i])
      # dscore433=dscore433+(B433[i]*(Xmat3[i,]%*%t(Xmat3[i,])))*((type4==1)[i])
    }
    dscore11=dscore111+dscore211+dscore311+dscore411
    dscore12=dscore112+dscore212+dscore312+dscore412
    dscore13=dscore113+dscore313
    dscore21=t(dscore12)
    dscore22=dscore122+dscore222+dscore322+dscore422
    dscore23=dscore123+dscore323
    dscore31=t(dscore13)
    dscore32=t(dscore23)
    dscore33=dscore133+dscore333
  }

  ##
  #The score function or the first derivative of the loglikelihood function:
  dscore=rbind(cbind(dscore11,dscore12,dscore13),cbind(dscore21,dscore22,dscore23),cbind(dscore31,dscor
  ##
  return(-dscore)
}

FreqID.LT.real.data <- function(Y, lin.pred, data, model = "semi-Markov", startVals, frailty=TRUE, meth
{

  ##
  y1     <- as.vector(Y[,1])
  delta1 <- as.vector(Y[,2])
  y2     <- as.vector(Y[,3])
  delta2 <- as.vector(Y[,4])
  l      <- as.vector(Y[,5])
  # Xmat1=as.matrix(data[,(6:(p+5))])
  # Xmat2=as.matrix(data[,(6:(p+5))])
  # Xmat3=as.matrix(data[,(6:(p+5))])
  Xmat1  <- as.matrix(model.frame(lin.pred[[1]], data=data))
  Xmat2  <- as.matrix(model.frame(lin.pred[[2]], data=data))
  Xmat3  <- as.matrix(model.frame(lin.pred[[3]], data=data))
```

```
##
fit.survreg.1 <- survreg(as.formula(paste("Surv(y1, delta1) ", as.character(lin.pred[[1]])[1], as.cha
fit.survreg.2 <- survreg(as.formula(paste("Surv(y2, delta2) ", as.character(lin.pred[[2]])[1], as.cha
# data.delta1_1 = data[delta1==1,]
# data.delta1_1$y2.m.y1 = y2[delta1==1] - y1[delta1==1]
data.delta1_1 = data[delta1==1,]
data.delta1_1$y2.m.y1 = y2[delta1==1] - y1[delta1==1]
data.delta1_1=data.delta1_1[-c(which(data.delta1_1$y2.m.y1==0)),]
fit.survreg.3 <- survreg(as.formula(paste("Surv(y2.m.y1, delta2) ", as.character(lin.pred[[3]])[1], as
alpha1      <- 1 / fit.survreg.1$scale
alpha2      <- 1 / fit.survreg.2$scale
alpha3        <- 1 / fit.survreg.3$scale

if (is.null(startVals)==T){
  startVals     <- c(-alpha1*coef(fit.survreg.1)[1], log(alpha1),
                     -alpha2*coef(fit.survreg.2)[1], log(alpha2),
                     -alpha3*coef(fit.survreg.3)[1], log(alpha3))
  if(frailty == TRUE) startVals <- c(startVals, 0.5)
  startVals     <- c(startVals,
                     -coef(fit.survreg.1)[-1] * alpha1,
                     -coef(fit.survreg.2)[-1] * alpha2,
                     -coef(fit.survreg.3)[-1] * alpha3)
}

if(model == "semi-Markov")
{
  if (method == "optim"){
    cat("Fitting illness-death model with Weibull baseline hazards ... this should take < 1 min \n")
    logLike <- function(p) logLike.weibull.SCR.SM.LT(p, y1=y1, y2=y2, delta1=delta1, delta2=delta2, l
                                                     Xmat1=Xmat1, Xmat2=Xmat2, Xmat3=Xmat3, frailty=fi
    optim.control = list(REPORT = 50)
    fit1 <- optim(startVals, #* runif(length(startVals), 0.9, 1.1),
                  logLike, hessian = TRUE, method="Nelder-Mead", control = optim.control)
    value <- list(estimate=fit1$par, H=fit1$hessian, logLike=-fit1$value, code=fit1$convergence)#, Xm
  }
  if (method == "nlm"){
    cat("Fitting illness-death model with Weibull baseline hazards ... this should take < 1 min \n")
    fit1  <- suppressWarnings(nlm(logLike.weibull.SCR.SM.LT, p=startVals,
                                  y1=y1, delta1=delta1, y2=y2, delta2=delta2, Xmat1=as.matrix(Xmat1),
                                  l = l, frailty=frailty,
                                  iterlim=1000, hessian=TRUE))
    value <- list(estimate=fit1$est, H=fit1$hessian, logLike=-fit1$minimum, code=fit1$code)
  }
}
##
if(model == "semi-Markov")
{
  class(value) <- c("Freq", "ID", "Ind", "WB", "semi-Markov")
}

return(value)
##
invisible()
```

```
}
ss2 <- function(j,tmpb,Q,B)
{
  a <- sum(tmpb*Q[,j])-tmpb[j]*Q[j,j]
  s <- 2*(a-B[j])
  return(s)
}

solveAdaLasso <- function(p,x,y,init,weight,lambda)
{
  Q = t(x)%*%x
  B = t(x)%*%y
  i=0
  status = 0

  lams =lambda*weight
  oldbeta <- init
  tmpbeta <- oldbeta

  while (i<150 && status==0){
    for (j in 1:p){
      s <- ss2 (j,tmpbeta,Q,B)
      if (s > lams[j])
        tmpbeta[j]<-(lams[j]-s)/(2*Q[j,j])
      else if (s < (-lams[j]) )
        tmpbeta[j]<-(-lams[j]-s)/(2*Q[j,j])
      else
        tmpbeta[j]<- 0.0
    }
    dx<-max(abs(tmpbeta-oldbeta))
    oldbeta <- tmpbeta
    if (dx<=1e-06)
      status <- 1
    i <- i+1
  }
  return(tmpbeta)
}

solveLasso <- function(yyy, XXX, lambda){
  n=nrow(XXX)
  p=ncol(XXX)

  #Step 1: initialize beta, using reg. least square:
  XXXprime=t(XXX)
  first=XXXprime%*%XXX + 2*lambda
  second=XXXprime%*%yyy
  # install.packages("pracma")
  library(pracma)
  beta <- mldivide(first,second)#vector of #23 elements

  #Step 2: for k=0,1,...,m, repeat:
  #convergence flag
  found <- 0
```

```
  # convergence tolerance
  TOL <- 1e-6
  while( found==0 ){
    #USing the current value of beta
    beta_old <- beta
    #optimize elements of beta by coordinate descent algorithm:
    for (i in 1:p){
      xxxi=XXX[,i]
      yyyi=yyy - XXX[,-i]%*%beta[-i,]
      xxxiprime=t(xxxi)
      deltai=xxxiprime%*%yyyi
      if (deltai< (-lambda)){
        firstt=deltai+lambda
        secondd=xxxiprime%*%xxxi
        beta[i]=firstt/secondd
      }
      else{
        if (deltai>lambda){
          firsttt=deltai-lambda
          seconddd=xxxiprime%*%xxxi
          beta[i]=firsttt/seconddd
        }
        else{
          beta[i]=0
        }
      }
      if (max(abs(beta-beta_old))<=TOL){
        found=1
      }
    }
  }
  #save outputs


  z.beta <- beta
  return(z.beta)


}

solveBAR <- function(Y, X, lambda,xi){

  p=ncol(X)
  #Step 1: initialize beta, using ridge reg:
  Im <- diag(1, p, p)
  beta <- solve(t(X)%*% X + xi*Im)%*%t(X)%*%Y
  #Step 2: for k=0,1,...,m, repeat:
  #convergence flag
  found <- 0
  # convergence tolerance
  TOL <- 1e-6
  count=0
  d <- 0.001 #to prevent computation overflow:
```

```
  while( found==0 ){
    #USing the current value of beta
    beta_old <- beta
    D <- diag( as.vector(1/(beta_old^2 + d^2)), p, p)
    beta <- solve(t(X)%*%X + lambda*D)%*%t(X)%*%Y
    count=count+1
    if (count>100){
      break;
    }
    if (max(abs(beta-beta_old))<=TOL){
      found=1
    }
  }
  #save outputs
  z.beta <- beta
  return(z.beta)
}

AdaLasso.finder.iter.solveAdaLasso=function(weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xm
  para.est=c(weibull.estimates,unpen.est)
  G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  X_irls=chol(H)
  Y_irls=forwardsolve(t(X_irls),H%*%unpen.est-G)

  beta=unpen.est
  print(beta)
  para.est=c(weibull.estimates,beta)
  flag=0
  TOL <- 1e-6
  count=0
  while (flag==0 && count<=100){
    betaold=beta
    para.est=c(weibull.estimates,betaold)
    G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    X_irls=chol(H)
    Y_irls=forwardsolve(t(X_irls),H%*%betaold-G)
    AdaLasso <-  solveAdaLasso((dim(X_irls)[1]),X_irls,Y_irls,unpen.est,1/abs(unpen.est),lam)

    beta=AdaLasso
    if (max(abs(beta-betaold))<=TOL){
      flag=1
    }
    count=count+1


  }

  return(list(beta=beta,H=H,G=G,X=X_irls,y=Y_irls,count=count))

}
```

```
lasso.finder.iter.solvelasso=function(weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat2,Xm
  para.est=c(weibull.estimates,unpen.est)
  G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  X_irls=chol(H)
  Y_irls=forwardsolve(t(X_irls),H%*%unpen.est-G)

  beta=unpen.est
  print(beta)
  # para.est=c(weibull.parameters,beta)
  para.est=c(weibull.estimates,beta)
  flag=0
  TOL <- 1e-6
  count=0
  while (flag==0 && count<=100){
    betaold=beta
    para.est=c(weibull.estimates,betaold)
    G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    X_irls=chol(H)
    Y_irls=forwardsolve(t(X_irls),H%*%betaold-G)
    lasso <-  solveLasso(Y_irls,X_irls,lam)

    beta=lasso
    if (max(abs(beta-betaold))<=TOL){
      flag=1
    }
    count=count+1


  }

  return(list(beta=beta,H=H,G=G,X=X_irls,y=Y_irls,count=count))

}

BAR.finder.iter.solveBAR=function(tol,weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat2,Xm
  para.est=c(weibull.estimates,unpen.est)
  G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
  X_irls=chol(H)
  Y_irls=forwardsolve(t(X_irls),H%*%unpen.est-G)

  beta=unpen.est
  print(beta)
  # para.est=c(weibull.parameters,beta)
  para.est=c(weibull.estimates,beta)
  flag=0
  TOL <- 1e-6
  count=0
  while (flag==0 && count<=100){
    betaold=beta
    para.est=c(weibull.estimates,betaold)
```

```
    G=dlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    H=ddlogLike.weibull.new(para.est, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xmat3, frailty=TRUE)
    X_irls=chol(H)
    Y_irls=forwardsolve(t(X_irls),H%*%betaold-G)
    bar <-  solveBAR(Y_irls,X_irls,lam,xi)


    beta=bar
    if (max(abs(beta-betaold))<=TOL){
      flag=1
    }
    count=count+1


  }
  cat("last one:\n")
  # print(abar)

  return(list(beta=beta,H=H,G=G,X=X_irls,y=Y_irls,count=count))

}

GCV.finder.AdaLasso=function(lambdavec,weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat2,
  for (lam in lambdavec){
    AdaLasso.est=AdaLasso.finder.iter.solveAdaLasso(weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l

    betavarsel.AdaLasso[,(which(lambdavec==lam))]=AdaLasso.est$beta
    betahat.and.weibull=c(weibull.estimates,AdaLasso.est$beta)

    G.tilde.AdaLasso=-ddlogLike.weibull.new(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat
    s.beta=c(rep(NA,length(AdaLasso.est$beta)))
    for (i in 1:length(AdaLasso.est$beta)){
      if (AdaLasso.est$beta[i]==0){
        s.beta[i]=0.000001
      }else{
        s.beta[i]=abs(AdaLasso.est$beta[i])
      }
    }
    n=dim(Xmat1)[1]
    A=diag(1/s.beta)
    first=solve(G.tilde.AdaLasso+(lam*A))

    p.lambda=sum(diag(first%*%G.tilde.AdaLasso))
    numerator=logLike.weibull.SCR.SM.LT(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xm
    denominator=n*(1-(p.lambda/n))^2
    GCV.AdaLasso[which(lambdavec==lam)]=numerator/ denominator
    cat("this is GCV for lambda :",lam,"==>",GCV.AdaLasso,"\n")
    optimal.gcv=which(GCV.AdaLasso==min(GCV.AdaLasso[!is.na(GCV.AdaLasso)]))
    beta.GCV=betavarsel.AdaLasso[,optimal.gcv]
  }
  GCV.AdaLasso=GCV.AdaLasso
  lam.final=lambdavec[optimal.gcv]
```

```
  return(list(GCV.AdaLasso=GCV.AdaLasso,lam.final=lam.final,optimal.gcv=optimal.gcv,beta.GCV=beta.GCV,G
}


GCV.finder.lasso=function(lambdavec,weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat2,Xma
  for (lam in lambdavec){
    lasso.est=lasso.finder.iter.solvelasso(weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xma

    betavarsel.lasso[,(which(lambdavec==lam))]=lasso.est$beta
    betahat.and.weibull=c(weibull.estimates,lasso.est$beta)

    G.tilde.lasso=-ddlogLike.weibull.new(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat2, X
    s.beta=c(rep(NA,length(lasso.est$beta)))
    for (i in 1:length(lasso.est$beta)){
      if (lasso.est$beta[i]==0){
        s.beta[i]=0.000001
      }else{
        s.beta[i]=abs(lasso.est$beta[i])
      }
    }
    n=dim(Xmat1)[1]
    A=diag(1/s.beta)
    first=solve(G.tilde.lasso+(lam*A))

    p.lambda=sum(diag(first%*%G.tilde.lasso))
    numerator=logLike.weibull.SCR.SM.LT(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xm
    denominator=n*(1-(p.lambda/n))^2
    GCV.lasso[which(lambdavec==lam)]=numerator/ denominator
    cat("this is GCV for lambda :",lam,"==>",GCV.lasso,"\n")
    optimal.gcv=which(GCV.lasso==min(GCV.lasso[!is.na(GCV.lasso)]))
    beta.GCV=betavarsel.lasso[,optimal.gcv]
  }
  GCV.lasso=GCV.lasso
  lam.final=lambdavec[optimal.gcv]

  return(list(GCV.lasso=GCV.lasso,lam.final=lam.final,optimal.gcv=optimal.gcv,beta.GCV=beta.GCV,GCV=GCV
}


GCV.finder.BAR=function(tol,lambdavec,xi,weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat
  for (lam in lambdavec){
    cat("this lam running:",lam,"\n")
    bar.est=BAR.finder.iter.solveBAR(tol,weibull.estimates,unpen.est,Y,y1,y2,delta1,delta2,l,Xmat1,Xmat

    betavarsel.bar[,(which(lambdavec==lam))]=bar.est$beta
    betahat.and.weibull=c(weibull.estimates,bar.est$beta)

    G.tilde.bar=-ddlogLike.weibull.new(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xma
    s.beta=c(rep(NA,length(bar.est$beta)))
    for (i in 1:length(bar.est$beta)){
      if (bar.est$beta[i]==0){
        s.beta[i]=0.000001
      }else{
```

```
        s.beta[i]=abs(bar.est$beta[i])
      }
    }
    n=dim(Xmat1)[1]
    A=diag(1/s.beta)
    first=solve(G.tilde.bar+(lam*A))

    p.lambda=sum(diag(first%*%G.tilde.bar))
    numerator=logLike.weibull.SCR.SM.LT(betahat.and.weibull, y1, y2, delta1, delta2, l, Xmat1, Xmat2, Xm
    denominator=n*(1-(p.lambda/n))^2
    GCV.bar[which(lambdavec==lam)]=numerator/ denominator
    cat("this is GCV for lambda :",lam,"==>",GCV.bar,"\n")
    optimal.gcv=which(GCV.bar==min(GCV.bar[!is.na(GCV.bar)]))
    beta.GCV=betavarsel.bar[,optimal.gcv]
  }
  GCV.bar=GCV.bar
  lam.final=lambdavec[optimal.gcv]

  return(list(GCV.bar=GCV.bar,lam.final=lam.final,optimal.gcv=optimal.gcv,beta.GCV=beta.GCV,GCV=GCV.bar
}



code_start_time<-Sys.time()
data<-colon
data<-data[!is.na(data[,9]),]
data<-data[!is.na(data[,11]),]
num_type=2
data_length<-length(data$id)
n<-data_length/2
maxiter=200
eps=1e-4
p=12
K=2
num_tuning=20
replicate=1
r=1
beta_hat_lasso_array<-array(0,dim=c(p,K,replicate))
beta_hat_lasso_adap_array<-array(0,dim=c(p,K,replicate))
beta_hat_origin_array<-array(0,dim=c(p,K,replicate))
beta_hat_array<-array(0,dim=c(p,K,replicate))
beta_hat_adap_array<-array(0,dim=c(p,K,replicate))
adapweight<-array(0,dim=c(p,K,replicate))


Data_analysis<-function(start,t,delta,x,adapweight_bi_level,tuning,maxiter,eps=1e-4,trace=FALSE){

  meanx<-matrix(0,p,K)
  normx<-matrix(0,p,K)
  covariate<-array(0,dim=c(n,p,K))
  x.gamma<-array(0,dim=c(n,p,K))
  beta.scaled<-matrix(0,p,K)
```

```
for(k in 1:K){
  covariate[,,k] = scale(x[,,k])
  meanx[,k] = attributes(scale(x[,,k]))$'scaled:center'
  normx[,k] = attributes(scale(x[,,k]))$'scaled:scale'
}


## initial value ##
gamma = rep(1,p)
gamma.old = gamma
theta<-matrix(0,p,K)
theta.old = matrix(0,p,K)
if(missing(adapweight_bi_level)){
  adapweight_bi_level=matrix(1,p,K)
}
iter_record<-0
iter = 0
dif = 1

while(iter<maxiter && dif>eps) {
  iter = iter + 1
  ## update theta ##
  for(k in 1:K){
    x.theta=t(t(covariate[,,k])*gamma)
    x.theta = scale(x.theta, FALSE, adapweight_bi_level[,k]) ## incorporate adaptive weights
    fit.theta = penalized(Surv(start,t[,k],delta[,k]), penalized=x.theta, unpenalized = ~0, model='co
    theta[,k] = as.vector(attributes(fit.theta)$penalized)/adapweight_bi_level[,k]
    x.gamma[,,k]= t(t(covariate[,,k])*theta[,k])
  }

  ## Update gamma Using Cycle Coordinate Descent##
  maxiter_gamma=maxiter
  eps_gamma=0.5
  iter_gamma=0
  dif_gamma=1
  gamma_origin=gamma
  gamma_update=gamma
  u_gamma<-rep(0,p)
  A_gamma<-rep(0,p)
  while(iter_gamma<maxiter_gamma && dif_gamma>eps_gamma){
    iter_gamma=iter_gamma+1
    for(j in 1:p){
      if(theta[j,1]==0&theta[j,2]==0){
        gamma_update[j]=0
        next
      }else{
        for(i in 1:n){
          for(k in 1:K){
            S_1=0
            S=0
            S_2=0
            for(m in 1:n){
              betax_l=gamma_origin%*%x.gamma[m,,k]
```

```
                S_1=ifelse(t[m,k]>=t[i,k],1,0)*exp(betax_l)*x.gamma[m,j,k]+S_1
                S=ifelse(t[m,k]>=t[i,k],1,0)*exp(betax_l)+S
                S_2=ifelse(t[m,k]>=t[i,k],1,0)*exp(betax_l)*((x.gamma[m,j,k])^2)+S_2
              }
              u_gamma[j]=u_gamma[j]+delta[i,k]*(x.gamma[i,j,k]-S_1/S)
              A_gamma[j]=A_gamma[j]+delta[i,k]*(-S_2/S+(S_1/S)^2)
            }
          }
          u_gamma[j]=u_gamma[j]
          A_gamma[j]=A_gamma[j]
          if(gamma_origin[j]>=0){
            gamma_update[j]=max(0,gamma_origin[j]-(u_gamma[j]-tuning)/A_gamma[j])
          }else{
            gamma_update[j]=min(0,gamma_origin[j]-(u_gamma[j]+tuning)/A_gamma[j])
          }
        }
      }
      dif_gamma=max(abs(gamma_update-gamma_origin))
      gamma_origin=gamma_update
      cat('iter_gamma:', iter_gamma, '\n')
      cat('dif_gamma:', dif_gamma, '\n')
    }
    gamma= as.vector(abs(gamma_origin))

    dif = max(max(abs(theta-theta.old)), max(abs(gamma-gamma.old)))
    theta.old = theta
    gamma.old = gamma

    cat('iter:', iter, '\n')
    cat('dif:', dif, '\n')
  }

  beta.scaled= theta*gamma
  beta_hat =beta.scaled/normx

  iter_record<-iter
  return(beta_hat)
}


t=matrix(0,data_length/num_type,num_type)
delta=matrix(0,data_length/num_type,num_type)
count1_t=1
count2_t=1
count1_delta=1
count2_delta=1
etype<-data$etype
for(i in 1:data_length){
  if(etype[i]==1){
    t[count1_t,1]=data$time[i]
    delta[count1_delta,1]=data$status[i]
    count1_t=count1_t+1
    count1_delta=count1_delta+1
```

```
  } else {

    t[count2_t,2]=data$time[i]
    delta[count2_delta,2]=data$status[i]
    count2_t=count2_t+1
    count2_delta=count2_delta+1
  }
}

######Generating Covariates#####
x<-array(0,dim=c(data_length/2,12,num_type))
rx<-data$rx
count1_lev=1
count2_lev=1
count1_lev5fu=1
count2_lev5fu=1
for(i in 1:data_length){
  if(etype[i]==1){
    if(rx[i]=="Lev"){
      x[count1_lev,1,1]=1
    } else {
      x[count1_lev,1,1]=0
    }
    if(rx[i]=="Lev+5FU"){
      x[count1_lev,2,1]=1
    } else {
      x[count1_lev,2,1]=0
    }
    count1_lev=count1_lev+1
    count1_lev5fu=count1_lev5fu+1
  } else {
    if(rx[i]=="Lev"){
      x[count1_lev,1,2]=1
    } else {
      x[count1_lev,1,2]=0
    }
    if(rx[i]=="Lev+5FU"){
      x[count1_lev,2,2]=1
    } else {
      x[count1_lev,2,2]=0
    }
    count2_lev=count2_lev+1
    count2_lev5fu=count2_lev5fu+1
  }
}

count1_sex=1
count2_sex=1
count1_age=1
count2_age=1
count1_obs=1
count2_obs=1
count1_per=1
```

```
count2_per=1
count1_adh=1
count2_adh=1
count1_nods=1
count2_nods=1
count1_dif=1
count2_dif=1
count1_ext=1
count2_ext=1
count1_sur=1
count2_sur=1
count1_nod4=1
count2_nod4=1
etype<-data$etype
for(i in 1:data_length){
  if(etype[i]==1){
    x[count1_sex,3,1]=data$sex[i]
    x[count1_age,4,1]=data$age[i]
    x[count1_obs,5,1]=data$obstruct[i]
    x[count1_per,6,1]=data$perfor[i]
    x[count1_adh,7,1]=data$adhere[i]
    x[count1_nods,8,1]=data$nodes[i]
    x[count1_dif,9,1]=data$differ[i]
    x[count1_ext,10,1]=data$extent[i]
    x[count1_sur,11,1]=data$surg[i]
    x[count1_nod4,12,1]=data$node4[i]
    count1_sex=count1_sex+1
    count1_age=count1_age+1
    count1_obs=count1_obs+1
    count1_per=count1_per+1
    count1_adh=count1_adh+1
    count1_nods=count1_nods+1
    count1_dif=count1_dif+1
    count1_ext=count1_ext+1
    count1_sur=count1_sur+1
    count1_nod4=count1_nod4+1
  } else {
    x[count2_sex,3,2]=data$sex[i]
    x[count2_age,4,2]=data$age[i]
    x[count2_obs,5,2]=data$obstruct[i]
    x[count2_per,6,2]=data$perfor[i]
    x[count2_adh,7,2]=data$adhere[i]
    x[count2_nods,8,2]=data$nodes[i]
    x[count2_dif,9,2]=data$differ[i]
    x[count2_ext,10,2]=data$extent[i]
    x[count2_sur,11,2]=data$surg[i]
    x[count2_nod4,12,2]=data$node4[i]
    count2_sex=count2_sex+1
    count2_age=count2_age+1
    count2_obs=count2_obs+1
    count2_per=count2_per+1
    count2_adh=count2_adh+1
    count2_nods=count2_nods+1
```

```
    count2_dif=count2_dif+1
    count2_ext=count2_ext+1
    count2_sur=count2_sur+1
    count2_nod4=count2_nod4+1
  }
}


meanx<-matrix(0,p,K)
normx<-matrix(0,p,K)
covariate<-array(0,dim=c(n,p,K))
for(k in 1:K){
  covariate[,,k] = scale(x[,,k])
  meanx[,k] = attributes(scale(x[,,k]))$'scaled:center'
  normx[,k] = attributes(scale(x[,,k]))$'scaled:scale'
}

Zcov=x[,,1]
Zcov=as.matrix(Zcov)


Zcovscale=covariate[,,1]
Zcovscale=as.matrix(Zcovscale)

start <- rep(0,n)
#first transition:
stop=t[,1]
time1=stop-start
status1=delta[,1]

#second transition:
stop=t[,2]
time2=stop-start
status2=delta[,2]


firstportionofdata=cbind(time1,status1,time2,status2)

dim(firstportionofdata)[1]==dim(Zcov)[1]
```

```
## [1] TRUE
```

```
dim(firstportionofdata)[1]==dim(Zcovscale)[1]
```

```
## [1] TRUE
```

```
#creating an artificial vector of left truncations all being zero:
L.crc=c(rep(0,dim(firstportionofdata)[1]))
dat.crc.reg=cbind(firstportionofdata,L.crc,Zcov)
dat.crc.scaled=cbind(firstportionofdata,L.crc,Zcovscale)

colnames(dat.crc.scaled)[1:17]=c("y1","delta1","y2","delta2","L","first","second","third","fourth","fif
```

```
colnames(dat.crc.reg)[1:17]=c("y1","delta1","y2","delta2","L","first","second","third","fourth","fifth"

lin.pred.crc= list(as.formula(~first+second+third+fourth+fifth+sixth+seventh+eighth+ninth+tenth+eleventl

dat.crc.reg=as.data.frame(dat.crc.reg)
dat.crc.scaled=as.data.frame(dat.crc.scaled)

#some data are wrongly entered so that y2-y1 for those for whom delta1 and delta2 are both 1 are
#the same. It should not be the case in semi-comp data. So, I dlete those cases which are 4 cases:

dat.crc.reg$del=dat.crc.reg$y2-dat.crc.reg$y1
indextodelete=which(dat.crc.reg$del==0&dat.crc.reg$delta1==1&dat.crc.reg$delta2==1)

dat.crc.reg=dat.crc.reg[-c(indextodelete),]
dat.crc.reg=dat.crc.reg[,-18]

dat.crc.scaled=dat.crc.scaled[-c(indextodelete),]
dat.crc.scaled=dat.crc.scaled[,-18]

dim(dat.crc.reg)
```

```
## [1] 884  17
```

```
dim(dat.crc.scaled)
```

```
## [1] 884  17
```

```
#unpenalized estimate with Weibull BH:
fit.wb.crc.reg=FreqID.LT.real.data(Y=dat.crc.reg, lin.pred=lin.pred.crc, data=dat.crc.reg, model = "semi
```

```
## Fitting illness-death model with Weibull baseline hazards ... this should take < 1 min
```

```
fit.wb.crc.scaled=FreqID.LT.real.data(Y=dat.crc.scaled, lin.pred=lin.pred.crc, data=dat.crc.scaled, mod
```

```
## Fitting illness-death model with Weibull baseline hazards ... this should take < 1 min
```

```
cbind(fit.wb.crc.reg$estimate,fit.wb.crc.scaled$estimate)
```

```
##                      [,1]         [,2]
## (Intercept)  -7.681316124 -6.061361814
##              -0.264568331 -0.232915987
## (Intercept) -10.901052726 -9.274211772
##              -0.016135959 -0.076481712
## (Intercept)  -8.412176726 -6.702873744
##               0.051823149  0.054258295
##              -0.027349017  0.097126885
## first         0.116772979  0.026673128
## second       -0.466712123 -0.213504308
## third        -0.107066598 -0.048546333
## fourth       -0.006621188 -0.004417372
```

```
## fifth          0.366540595  0.129376283
## sixth          0.316067773  0.078462662
## seventh        0.246694753  0.130777538
## eighth         0.056926216  0.226590086
## ninth          0.151109473  0.110562507
## tenth          0.511333436  0.241487686
## eleventh       0.276600543  0.142153949
## twelfth        0.716404280  0.323874320
## first         -0.215256969 -0.189909431
## second        -0.238400311 -0.057181335
## third          0.330930610  0.142942753
## fourth        -0.008539776  0.749720762
## fifth          0.429175502  0.126458776
## sixth          0.100097366 -0.102812029
## seventh        0.325941881  0.099160978
## eighth        -0.044555524  0.190296835
## ninth          0.111809986  0.189128442
## tenth          0.365911677  0.337807068
## eleventh       0.200457723  0.222383473
## twelfth        0.627496259  0.318667784
## first          0.177382280  0.095621271
## second         0.381029995  0.194712424
## third          0.294743808  0.151166869
## fourth         0.008852139  0.196727928
## fifth          0.330507358  0.153297510
## sixth         -0.345138355 -0.043499436
## seventh        0.061620163  0.128511777
## eighth         0.031518577  0.143372758
## ninth          0.018666744  0.060287045
## tenth          0.129746213  0.132365737
## eleventh       0.058770509  0.048881478
## twelfth        0.538628878  0.285122722
```

```
y1=dat.crc.reg$y1
y2=dat.crc.reg$y2
delta1=dat.crc.reg$delta1
delta2=dat.crc.reg$delta2
lin.pred=lin.pred.crc
data=dat.crc.reg

Y=data
head(data)
```

```
##     y1 delta1   y2 delta2 L first second third fourth fifth sixth seventh
## 1  968      1 1521      1 0     0      1     1     43     0     0       0
## 2 3087      0 3087      0 0     0      1     1     63     0     0       0
## 3  542      1  963      1 0     0      0     0     71     0     0       1
## 4  245      1  293      1 0     0      1     0     66     1     0       0
## 5  523      1  659      1 0     0      0     1     69     0     0       0
## 6  904      1 1767      1 0     0      1     0     57     0     0       0
##   eighth ninth tenth eleventh twelfth
## 1      5     2     3        0       1
## 2      1     2     3        0       0
## 3      7     2     2        0       1
```

```
## 4      6     2     3        1       1
## 5     22     2     3        1       1
## 6      9     2     3        0       1
```

```
#censoring rates:
censoring.observ.rates=matrix(NA,1,4)

censoring.observ.rates[1]=length(which(data$delta1==0&data$delta2==0))
censoring.observ.rates[2]=length(which(data$delta1==0&data$delta2==1))
censoring.observ.rates[3]=length(which(data$delta1==1&data$delta2==0))
censoring.observ.rates[4]=length(which(data$delta1==1&data$delta2==1))

censoring.observ.rates=as.data.frame(censoring.observ.rates)
colnames(censoring.observ.rates)=c("00","01","10","11")
censoring.observ.rates
```

```
##    00 01 10  11
## 1 405 37 53 389
```

```
# VARSEL-WEIBULL -----------------------------------------------------------


lambdavec.bar=seq(3,3.1,0.05)
lambdavec.lasso=seq(5.5,6.5,0.05)
lambdavec.ada=seq(5.5,6.5,0.05)
tol=1e-04
xi=10
nuisance.estimates.s=fit.wb.crc.reg$estimate[1:7]
unpen.est.s=fit.wb.crc.reg$estimate[8:length(fit.wb.crc.reg$estimate)]
Xmat1.s=Xmat2.s=Xmat3.s=as.matrix(data[,c(6:dim(data)[2])])

p=12
GCV.bar=vector()
GCV.selected.bar=matrix(0,length(lambdavec.bar),1)
betavarsel.bar=matrix(0,3*p,length(lambdavec.bar))
beta.selected.bar=matrix(0,3*p,1)
GCV.lasso=vector()
GCV.selected.lasso=matrix(0,length(lambdavec.lasso),1)
betavarsel.lasso=matrix(0,3*p,length(lambdavec.lasso))
beta.selected.lasso=matrix(0,3*p,1)
GCV.AdaLasso=vector()
GCV.selected.ada=matrix(0,length(lambdavec.ada),1)
betavarsel.AdaLasso=matrix(0,3*p,length(lambdavec.ada))
beta.selected.ada=matrix(0,3*p,1)


#BAR:
a.bar.weibull=GCV.finder.BAR(tol,lambdavec.bar,xi,nuisance.estimates.s,unpen.est.s,Y,data$y1,data$y2,da
```

```
## this lam running: 3
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
```

```
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## last one:
## this is GCV for lambda : 3 ==> 8.138422
## this lam running: 3.05
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## last one:
## this is GCV for lambda : 3.05 ==> 8.138422 8.139158
## this lam running: 3.1
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## last one:
## this is GCV for lambda : 3.1 ==> 8.138422 8.139158 8.139838
```

```
a.bar.weibull$betavarsel.bar->crc.bar.scale.res
crc.bar.scale.res=as.data.frame(crc.bar.scale.res)
rownames(crc.bar.scale.res)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","Nodes1","
colnames(crc.bar.scale.res)=lambdavec.bar
Answer.scaled.bar.GCV=a.bar.weibull$beta.GCV
Answer.scaled.bar.GCV=as.data.frame(Answer.scaled.bar.GCV)
rownames(Answer.scaled.bar.GCV)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","Nodes
indextobezeroinbar=which(abs(Answer.scaled.bar.GCV)<=tol)
Answer.scaled.bar.GCV[indextobezeroinbar,]=0

#LASSO:
b.lasso.weibull=GCV.finder.lasso(lambdavec.lasso,nuisance.estimates.s,unpen.est.s,Y,data$y1,data$y2,data
```

```
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.5 ==> 9.205591
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.55 ==> 9.205591 9.999539
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.6 ==> 9.205591 9.999539 3.251035
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.65 ==> 9.205591 9.999539 3.251035 7.307247
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
```

```
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.7 ==> 9.205591 9.999539 3.251035 7.307247 7.953452
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.75 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.8 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.85 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.5
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
```

```
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.9 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.95 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.53
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.52711
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.05 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.53
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
```

```
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.1 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.15 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.51
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.2 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.25 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.51
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
```

```
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.3 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.35 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.53
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.4 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.45 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.53
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
```

```
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.5 ==> 9.205591 9.999539 3.251035 7.307247 7.953452 8.218417 8.387325 8.527
```

```
b.lasso.weibull$betavarsel.lasso->crc.lasso.scale.res
crc.lasso.scale.res=as.data.frame(crc.lasso.scale.res)
rownames(crc.lasso.scale.res)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","Nodes1"
colnames(crc.lasso.scale.res)=lambdavec.lasso
Answer.scaled.lasso.GCV=b.lasso.weibull$beta.GCV
Answer.scaled.lasso.GCV=as.data.frame(Answer.scaled.lasso.GCV)
rownames(Answer.scaled.lasso.GCV)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","Nod

#ADALASSO:
c.adalasso.weibull=GCV.finder.AdaLasso(lambdavec.ada,nuisance.estimates.s,unpen.est.s,Y,data$y1,data$y2
```

```
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.5 ==> 7.800768
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.55 ==> 7.800768 7.882779
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.6 ==> 7.800768 7.882779 7.940649
```

```
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.65 ==> 7.800768 7.882779 7.940649 7.984518
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.7 ==> 7.800768 7.882779 7.940649 7.984518 8.019458
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.75 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.8 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
```

```
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.85 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.09
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.9 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 5.95 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.09
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.0944
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
```

```
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.05 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.0
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.1 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.15 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.0
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.2 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
```

```
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.25 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.09
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.3 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.35 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.09
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.4 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
```

```
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.45 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.09
##        first       second        third       fourth        fifth        sixth
##  0.116772979 -0.466712123 -0.107066598 -0.006621188  0.366540595  0.316067773
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.246694753  0.056926216  0.151109473  0.511333436  0.276600543  0.716404280
##        first       second        third       fourth        fifth        sixth
## -0.215256969 -0.238400311  0.330930610 -0.008539776  0.429175502  0.100097366
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.325941881 -0.044555524  0.111809986  0.365911677  0.200457723  0.627496259
##        first       second        third       fourth        fifth        sixth
##  0.177382280  0.381029995  0.294743808  0.008852139  0.330507358 -0.345138355
##      seventh       eighth        ninth        tenth     eleventh      twelfth
##  0.061620163  0.031518577  0.018666744  0.129746213  0.058770509  0.538628878
## this is GCV for lambda : 6.5 ==> 7.800768 7.882779 7.940649 7.984518 8.019458 8.048343 8.072957 8.094
```

```
c.adalasso.weibull$betavarsel.AdaLasso->crc.adalasso.scale.res
crc.adalasso.scale.res=as.data.frame(crc.adalasso.scale.res)
rownames(crc.adalasso.scale.res)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","Nodes
colnames(crc.adalasso.scale.res)=lambdavec.lasso
Answer.scaled.adalasso.GCV=c.adalasso.weibull$beta.GCV
Answer.scaled.adalasso.GCV=as.data.frame(Answer.scaled.adalasso.GCV)
rownames(Answer.scaled.adalasso.GCV)=c("Lev1","Lev+FU1","Sex1","Age1","Obstruct1","Perfor1","Adhere1","N

cbind(unpen.est.s,Answer.scaled.lasso.GCV ,Answer.scaled.adalasso.GCV ,Answer.scaled.bar.GCV)
```

```
##             unpen.est.s Answer.scaled.lasso.GCV Answer.scaled.adalasso.GCV
## Lev1        0.116772979             0.000000000                0.000000000
## Lev+FU1    -0.466712123            -0.485316075               -0.487593269
## Sex1       -0.107066598            -0.138325369                0.000000000
## Age1       -0.006621188            -0.004679875                0.000000000
## Obstruct1   0.366540595             0.147873899                0.108031984
## Perfor1     0.316067773             0.000000000                0.000000000
## Adhere1     0.246694753             0.054923998                0.000000000
## Nodes1      0.056926216             0.065205121                0.024065874
## Differ1     0.151109473             0.138809004                0.000000000
## Extent1     0.511333436             0.549506591                0.563501150
## Surg1       0.276600543             0.158298317                0.082759973
## Node41      0.716404280             0.515096508                0.824367829
## Lev2       -0.215256969             0.000000000                0.000000000
## Lev+FU2    -0.238400311             0.000000000                0.000000000
## Sex2        0.330930610             0.000000000                0.000000000
## Age2       -0.008539776             0.031527763                0.008976262
## Obstruct2   0.429175502             0.000000000                0.000000000
## Perfor2     0.100097366             0.000000000                0.000000000
## Adhere2     0.325941881             0.000000000                0.000000000
## Nodes2     -0.044555524             0.070949862                0.000000000
## Differ2     0.111809986            -0.249690255                0.000000000
## Extent2     0.365911677            -0.201874706                0.144615973
## Surg2       0.200457723             0.000000000                0.000000000
## Node42      0.627496259             0.000000000                0.058627677
```

```
## Lev3       0.177382280            0.000000000            0.000000000
## Lev+FU3    0.381029995            0.149968572            0.115506266
## Sex3       0.294743808            0.156858411            0.149224246
## Age3       0.008852139            0.012402675            0.013499565
## Obstruct3  0.330507358            0.114118408            0.056749347
## Perfor3   -0.345138355            0.000000000            0.000000000
## Adhere3    0.061620163            0.000000000            0.000000000
## Nodes3     0.031518577            0.056635429            0.003140042
## Differ3    0.018666744            0.004695171            0.000000000
## Extent3    0.129746213            0.137343154            0.154883721
## Surg3      0.058770509            0.000000000            0.000000000
## Node413    0.538628878            0.377106216            0.738429679
##           Answer.scaled.bar.GCV
## Lev1                 0.00000000
## Lev+FU1             -0.51292746
## Sex1                 0.00000000
## Age1                 0.00000000
## Obstruct1            0.00000000
## Perfor1              0.00000000
## Adhere1              0.00000000
## Nodes1               0.00000000
## Differ1              0.00000000
## Extent1              0.59446176
## Surg1                0.00000000
## Node41               0.99674734
## Lev2                 0.00000000
## Lev+FU2              0.00000000
## Sex2                 0.00000000
## Age2                 0.01655415
## Obstruct2            0.00000000
## Perfor2              0.00000000
## Adhere2              0.00000000
## Nodes2               0.00000000
## Differ2              0.00000000
## Extent2              0.00000000
## Surg2                0.00000000
## Node42               0.00000000
## Lev3                 0.00000000
## Lev+FU3              0.00000000
## Sex3                 0.00000000
## Age3                 0.02275525
## Obstruct3            0.00000000
## Perfor3              0.00000000
## Adhere3              0.00000000
## Nodes3               0.00000000
## Differ3              0.00000000
## Extent3              0.00000000
## Surg3                0.00000000
## Node413              0.81282141
```