Appendix IV

S-PLUS functions for the semiparametric approach

The functions listed and described in the following pages correspond to the implementation in S-PLUS of the semiparametric method introduced in Chapter 6. These functions have been used for the analysis of the HIV+PTB dataset included in Appendix I, as well as for the sensitivity analysis for this dataset, and for the simulation study carried out in Chapter 7.

In order to facilitate the reading of the functions, we include the meaning of the main variables (constants, vectors, matrices or lists) used in the implementation.

For ease of location, the names of the functions and the name of the objects appear in alphabetical order.

Name	:	**Short description of the function**
`beta()`	:	Computes the vector $\boldsymbol{\beta}$
`cGami()`	:	Computes the matrix $\boldsymbol{\Gamma}^{-1}$ in the estimation of the asymptotic variance-covariance matrix
`cLam1()`	:	Idem for the matrix $\boldsymbol{\Lambda}_1$
`cLam2()`	:	Idem for the matrix $\boldsymbol{\Lambda}_2$
`cOme()`	:	Idem for the matrix $\boldsymbol{\Omega}$
`cSTG()`	:	Idem for the matrix $\mathcal{STG}$
`dsn()`	:	Performs the simulations for each scenario
`dssse()`	:	Summarizes the simulations for each scenario
`EGKM()`	:	Computes the life table for the Estimated Grouped Kaplan–Meier estimator
`expit()`	:	Computes the inverse of the logit function
`gendata()`	:	Generates the data for the simulation for each scenario
`group()`	:	Classifies the observed survival time, Y, according to the partition $\{\tau_k\}_{k=0,\dots,K+1}$
`ind()`	:	Returns the index in $\mathcal{J}$ of an observation $(Y_i, \delta_i)'$
`norma()`	:	Returns the norm of a vector
`phi2()`	:	Computes the function $\phi^{(2)}(.) = (1, Y, Y^2)'$
`sa()`	:	Performs the sensitivity analysis for the HIV+PTB cohort
`scalpha()`	:	Returns the estimation of $\boldsymbol{\alpha}$ in the non-response probability pattern for a fixed value of τ
`se()`	:	Returns the vector of standard errors for the EGKM estimator
`Ui()`	:	Computes the contribution of the i-th individual to the estimating equations

Name	:	**Description of the object**
`alphao, alphan`	:	Vector of estimates for $\boldsymbol{\alpha}$
`beta`	:	Vector of estimates for $\boldsymbol{\beta}$
`C4`	:	Covariate $CD4$ of the HIV+PTB cohort (dichotomized)
`CC`	:	Censoring time vector in the simulation
`delta, deltaTB`	:	Censoring indicator vector
`K`	:	Number of knots in the partition $\{\tau_k\}_{k=1,\dots,K}$
`KMlist`	:	Life table resulting from the EGKM estimator
`mcaranal`	:	Logical value indicating the MCAR analysis
`n`	:	Sample size
`p`	:	$P(X=1)$
`PP`	:	Covariate PPD of the HIV+PTB cohort
`R, RC4, RPP`	:	Indicator vectors of response to $X, CD4$ and PPD
`simul`	:	List with the result of the simulations
`sp`	:	Estimated vector of non-response probabilities
`ssimul`	:	List with the summary of the simulations
`SV`	:	Observed survival time of the HIV+PYB cohor
`talpha`	:	True vector $\boldsymbol{\alpha}$ in the simulation
`tau, taua`	:	True τ and analyzing τin the non-response pattern in the simulation
`tmax`	:	Maximum value of the observation window
`tsp`	:	True vector of non-response probabilities
`TT`	:	Time to event vector in the simulation
`ttau`	:	Vector of the knots in the partition $\{\tau_k\}_{k=1,\dots,K}$
`x`	:	Value of the covariate X
`X`	:	Observed covariate vector X
`xop`	:	Value for selecting the covariate $CD4$ or PPD
`XX`	:	True covariate vector in the simulation
`Y, YY`	:	Observed time vector in the simulation

Function `beta(x)`: Computation of the vector `beta` for the category `x` from the life table `KMlist`

```
"beta"<-function(x = 0)
{
     if(x == 0) KMt <- KMlist$X0 else KMt <- KMlist$X1
     beta <- c(as.vector(t(KMt[1:K, 2:3])), KMt[K + 1, 3])
     beta
}
```

Function `cGami(x)`: Computation of the matrix $\mathbf{\Gamma}^{-1}$ in the estimation of the asymptotic variance-covariance matrix of the EGKM estimator for the category `x`. We choose $\phi(.) = (0, \dots, 0, 1)'$ if we are performing the MCAR case, or $\phi(.) = (0, \dots, 0, 1, Y, Y^2)'$ otherwise; where $\mathbf{\Gamma} = \begin{pmatrix} \text{diag}(\boldsymbol{d}) & \boldsymbol{B} \\ \mathbf{0} & m\boldsymbol{C} \end{pmatrix}$ and the function computes $\boldsymbol{d}$, $\boldsymbol{B}$ and $m\boldsymbol{C}$

```
"cGami"<-function(x = 0)
{
     beta <- beta(x)
     d <- rep(0, 2 * K + 1)
     B <- matrix(rep(0, (2 * K + 1) * dimq), ncol = dimq)
     mC <- matrix(rep(0, dimq^2), ncol = dimq)
     for(i in 1:n) {
          j <- ind(Y[i], delta[i], ttau, tmax)
          d[j] <- d[j] - R[i]/sp[i]
          if(mcaranal) {
               vec <- 1
               mat <- 1
          }
          else {
            vec <- c(1, Y[i], delta[i])
            mat <- matrix(c(1, Y[i], Y[i]^2, Y[i], Y[i]^2, Y[i]^3,
                delta[i], delta[i] * Y[i], delta[i] * Y[i]^2),
                ncol = dimq)
        }
        if(R[i] == 1) {
            B[j, ] <- B[j, ] + (R[i] * (m[j] * I(X[i] == x) - beta[j])
                * (-1) * (1 - sp[i]))/sp[i] * vec
            mC <- mC + (R[i] * (1 - sp[i]))/sp[i] * mat
        }
    }
    di <- 1/d
    di[di == Inf] <- 0
    if(mcaranal) {
        mCi <- as.numeric(1/mC)
        Bi <- - (di) * B * mCi
    }
    else {
        if(!(prod(eigen(mC)$values) == 0))
            mCi <- solve.Matrix(mC)
        else {
            mCi <- diag(dimq)
            errinv[ite] <<- 1
        }
        Bi <- - diag(di) %*% B %*% mCi
    }
    n * rbind(cbind(diag(di), Bi), cbind(matrix(rep(0, dimq * (2*K+1)),
        ncol = 2 * K + 1), mCi))
}
```

Function `cLam1(x)`: Computation of the matrix $\mathbf{\Lambda}_1$ in the estimation of the asymptotic variance-covariance matrix of the EGKM estimator for the category `x`

```
"cLam1"<-function(x = 0)
{
     M <- cGami(x)
     (M %*% cOme(x) %*% t(M))[1:(2 * K + 1), 1:(2 * K + 1)]
}
```

Function `cLam2(x)`: Computation of the matrix $\mathbf{\Lambda}_2$ in the estimation of the asymptotic variance-covariance matrix of the EGKM estimator for the category `x`

```
"cLam2" <- function(x = 0)
{
     if(x == 0) KMt <- KMlist$X0 else KMt <- KMlist$X1
     nefx <- KMlist$nef[x + 1]
     nef <- KMlist$nef[3]
     beta <- beta(x)
     dvalue <- (beta^2 * nef)/m
     dvalue[m == 0] <- 0
     Y <- group(YY, ttau, tmax)
     dw <- rep(0, 2 * K + 1)
     for(k in 1:K) {
          for(u in 1:0) {
               dw[2*k-u] <- sum(R/sp * I(Y == ttau[k]) * I(delta == u))
          }
     }
     dw[2 * K + 1] <- sum(R/sp * I(Y == tmax))
     w <- rep(0, 2 * K + 1)
     for(k in 1:K) {
          for(u in 1:0) {
            w[2*k-u] <- sum(R/sp * I(Y == ttau[k]) * I(delta == u) *
                I(X == x), na.rm = T)/dw[2 * k - u]
        }
    }
    w[2 * K + 1] <- sum(R/sp * I(Y == tmax) * I(X == x), na.rm = T)/
        dw[2 * K + 1]
    sigma2 <- rep(0, 2 * K + 1)
    for(k in 1:K) {
        for(u in 1:0) {
            sigma2[2 * k - u] <- sum(R/sp^2 * I(Y == ttau[k]) * I(delta
                == u) * (I(X == x) - w[2 * k - u])^2, na.rm = T)
        }
    }
    sigma2[2 * K + 1] <- sum(R/sp^2 * I(Y == tmax) * (I(X == x) -
        w[2 * K + 1])^2, na.rm = T)
    sigma2 <- n * dw^(-2) * sigma2
    v2 <- sum((beta^2 * (1 - m))/m + m^2 * sigma2, na.rm = T)/nefx^2
    diag(dvalue) - (beta %*% t(beta)) + diag(beta^2) * v2
}
```

Function cOme(x): Computation of the matrix $\mathbf{\Omega}$ in the estimation of the asymptotic variance-covariance matrix of the EGKM estimator for the category x

```
"cOme"<-function(x = 0)
{
    Ome <- matrix(rep(0, (2*K + 1 + dimq)^2), ncol = (2*K + 1 + dimq))
    for(i in 1:n)
        Ome <- Ome + Ui(i, x) %*% t(Ui(i, x))
    Ome/n
}
```

Function `cSTG(x)`: Computation of the matrix $\mathcal{STG}$ in the estimation of the asymptotic variance-covariance matrix of the EGKM estimator for the category `x`

```
"cSTG"<-function(x = 0)
{
     if(x == 0) KMt <- KMlist$X0 else KMt <- KMlist$X1
     S <- diag(KMt[1:K, 4])
     mT <- diag(K)
     for(i in 2:K)
          mT[i, 1:(i - 1)] <- 1
     G <- matrix(0, K, 2 * K + 1)
     for(i in 1:K) {
          G[i, 2 * i - 1] <- -1/KMt[i, 1]
          G[i, (2 * i):(2 * K + 1)] <- KMt[i, 2]/(KMt[i, 1] * (KMt[i, 1]
               - KMt[i, 2]))
     }
     S %*% mT %*% G
}
```

Function `dsn(previous, miter, optmax, opgrid, opsize, opprop, oppatt, opanal)`: After `previous` initial realizations, the function returns a list with the `miter` subsequent realizations of the simulation of the scenario defined by `optmax`, `opgrid`, `opsize` and `opprop` with data generated by an `oppatt` pattern and analyzed with an `opanal` pattern

```
"dsn"<-function(previous = 0, miter = 1000, optmax = 1, opgrid = 1,
     opsize = 1, opprop = 1, oppatt = 1, opanal = 1)
{
     library(Matrix)
     set.seed(157)
```

```
simul <<- list(pXX1 = rep(0, miter), pC = rep(0, miter),
     pM = rep(0, miter), nef = rep(0, miter), pX1 = rep(0, miter),
     SX0 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)),
     seSX0 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)),
     pc0 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)),
     SX1 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)),
     seSX1 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)),
     pc1 = structure(.Data=rep(0,miter*2*optmax),.Dim=c(miter,2*optmax)))
errinv <<- rep(0, miter)
TSX0 <- 1 - pweibull(c(1, 2, 5, 8, 15, 25) * 365, 1/1.4, exp(6.7))
TSX1 <- 1 - pweibull(c(1, 2, 5, 8, 15, 25) * 365, 1/0.8, exp(7.7))
tmax <<- 365 * switch(optmax,
     3,
     10,
     30)
if(optmax == 1)
     ttau <<- switch(opgrid,
          (1:2) * 365,
          (1:24) * 30,
          (1:104) * 7)
else ttau <<- switch(opgrid,
          (1:8) * 365,
          (1:96) * 30,
          (1:416) * 7)
K <<- length(ttau)
indti <- 1:(2 * optmax)
if(optmax == 1)
     tint <- switch(opgrid,
          c(1, 2),
          c(12, 24),
          c(52, 104))
else tint <- switch(opgrid,
          c(1, 2, 5, 8),
          c(12, 24, 60, 96),
         c(52, 104, 260, 416))
n <<- ifelse(optmax == 1, switch(opsize,
     100,
     500,
     1000), switch(opsize,
     200,
     1000,
     2000))
p <- switch(opprop,
     0.3,
     0.5)
talpha <- switch(oppatt,
     c(1, 0, 0),
     c(-0.75, 0.005, 1),
     c(-0.75, 0.005, 1),
     c(-0.75, 0.005, 1))
tau <- switch(oppatt,
     0,
     0,
     -2,
     2)
taua <- switch(opanal,
     0,
     0,
     0,
     -2,
     -1,
     1,
     2)
if(opanal == 2) {
     mcaranal <<- TRUE
     dimq <<- 1
}
else {
     mcaranal <<- FALSE
     dimq <<- 3
}
if(previous > 0)
     for(i in 1:previous)
         gendata(tmax, p, n, talpha, tau, , )
for(ite in 1:miter) {
     print(ite)
     gendata(tmax, p, n, talpha, tau, , )
     simul$pXX1[ite] <<- sum(XX)/n
     simul$pC[ite] <<- 1 - sum(delta)/n
     simul$pM[ite] <<- 1 - sum(R)/n
     if(opanal == 1) {
         Y <<- Y[R == 1]
         delta <<- delta[R == 1]
         X <<- X[R == 1]
         KMlist <<- survfit(Surv(Y, delta) ~ X, conf.type = "plain")
         rang0 <- 1:KMlist$strata[1]
         rang1 <- (KMlist$strata[1]+1):(KMlist$strata[1]+KMlist$strata[2])
         simul$nef[ite] <<- sum(R)
         simul$pX1[ite] <<- sum(X)/sum(R)
         tint0 <- c(sum(KMlist$time[rang0] <= 365), sum(KMlist$time[rang0]
             <= 2 * 365))
         tint1 <- c(sum(KMlist$time[rang1] <= 365), sum(KMlist$time[rang1]
             <= 2 * 365))
         if(optmax == 2) {
             tint0 <- c(tint0, sum(KMlist$time[rang0] <= 5 * 365),
                 sum(KMlist$time[rang0] <= 8 * 365))
             tint1 <- c(tint1, sum(KMlist$time[rang1] <= 5 * 365),
                 sum(KMlist$time[rang1] <= 8 * 365))
         }
         tint1 <- KMlist$strata[1] + tint1
         simul$SX0[ite, ] <<- KMlist$surv[tint0]
         simul$seSX0[ite, ] <<- KMlist$surv[tint0]*KMlist$std.err[tint0]
         simul$SX1[ite, ] <<- KMlist$surv[tint1]
         simul$seSX1[ite, ] <<- KMlist$surv[tint1]*KMlist$std.err[tint1]
     }
     else {
         scalpha(, , , taua)
         EGKM()
         simul$nef[ite] <<- KMlist$nef[3]
         simul$pX1[ite] <<- KMlist$nef[2]/KMlist$nef[3]
         simul$SX0[ite, ] <<- KMlist$X0[tint, 4]
         simul$seSX0[ite, ] <<- se(0)[tint]
         simul$SX1[ite, ] <<- KMlist$X1[tint, 4]
         simul$seSX1[ite, ] <<- se(1)[tint]
     }
     simul$pc0[ite, ] <<- as.numeric(I(TSX0[indti]>(simul$SX0[ite,]
         - 1.96*simul$seSX0[ite, ]))) * as.numeric(I(TSX0[indti]
         <(simul$SX0[ite, ] + 1.96*simul$seSX0[ite, ])))
     simul$pc1[ite, ] <<- as.numeric(I(TSX1[indti]>(simul$SX1[ite,]
         - 1.96*simul$seSX1[ite, ]))) * as.numeric(I(TSX1[indti]
         <(simul$SX1[ite, ] + 1.96*simul$seSX1[ite, ])))
     print(ite)
}
if(sum(errinv) == 0)
     print("OK")
else print(sum(errinv))
simul
}
```

Function `dssse(optmax, simul)`: Provides a list that summarizes the realizations of the simulations in `simul`

```
"dssse"<-function(optmax, simul)
{
    ssimul <- list(pXX1 = 0, pC = 0, pM = 0, nef = 0, pX1 = 0,
        SX0 = rep(0, 2 * optmax), medseSX0 = rep(0, 2 * optmax),
        pc0 = rep(0, 2 * optmax), rb0 = rep(0, 2 * optmax),
        sseSX0 = rep(0, 2 * optmax), SX1 = rep(0, 2 * optmax),
        medseSX1 = rep(0, 2 * optmax), pc1 = rep(0, 2 * optmax),
        rb1 = rep(0, 2 * optmax), sseSX1 = rep(0, 2 * optmax))
    TSX0 <- 1 - pweibull(c(1, 2, 5, 8, 15, 25) * 365, 1/1.4, exp(6.7))
    TSX1 <- 1 - pweibull(c(1, 2, 5, 8, 15, 25) * 365, 1/0.8, exp(7.7))
    indti <- 1:(2 * optmax)
    ssimul$pXX1 <- mean(simul$pXX1)
    ssimul$pC <- mean(simul$pC)
    ssimul$pM <- mean(simul$pM)
    ssimul$nef <- mean(simul$nef)
    ssimul$pX1 <- mean(simul$pX1)
    ssimul$SX0 <- apply(simul$SX0, 2, mean)
    for(i in indti) ssimul$medseSX0[i] <- location.lms(simul$seSX0[,i])$loc
    ssimul$pc0 <- apply(simul$pc0, 2, mean, na.rm = T)
    ssimul$rb0 <- (ssimul$SX0 - TSX0[indti])/TSX0[indti]
    ssimul$sseSX0 <- sqrt(apply(simul$SX0, 2, var))
    ssimul$SX1 <- apply(simul$SX1, 2, mean)
    for(i in indti) ssimul$medseSX1[i] <- location.lms(simul$seSX1[,i])$loc
    ssimul$pc1 <- apply(simul$pc1, 2, mean, na.rm = T)
    ssimul$rb1 <- (ssimul$SX1 - TSX1[indti])/TSX1[indti]
    ssimul$sseSX1 <- sqrt(apply(simul$SX1, 2, var))
    ssimul
}
```

Function `EGKM(tracta.NA, mcar)`: Computation of the Estimated Grouped Kaplan–Meier estimator from the semiparametric estimating equations. The function allows to treat NA values as well as the MCAR analysis. Results are summarized in a life table `KMlist`

```
"EGKM"<-function(tracta.NA = TRUE, mcar = FALSE)
{
     Y <- group(YY, ttau, tmax)
     KMlistt <- list(t = c(ttau, tmax),
          R1 = structure(.Data = rep(NA, 2*(K+1)), .Dim = c(K+1, 2)),
          R0 = structure(.Data = rep(NA, 2*(K+1)), .Dim = c(K+1, 2)),
          X0 = structure(.Data = c(rep(0, K), NA, rep(0, K), NA,
          rep(0, K + 1), rep(0, K), NA), .Dim = c(K + 1, 4)),
          X1 = structure(.Data = c(rep(0, K), NA, rep(0, K), NA,
          rep(0, K + 1), rep(0, K), NA), .Dim = c(K + 1, 4)),
          nef = rep(NA, 3))
     if((!(tracta.NA)) || mcar)
          sp <- 1
     for(k in 1:K) {
          for(j in 0:1) {
               KMlistt$R1[k, j + 1] <- sum(R*delta^(1-j)*(1-delta)^j*
                    I(Y == ttau[k]))
               KMlistt$R0[k, j + 1] <- sum((1-R)*delta^(1-j)*(1-delta)^j*
                    I(Y == ttau[k]))
               m <- KMlistt$R1[k, j + 1] + KMlistt$R0[k, j + 1] *
                    as.integer(tracta.NA)
               if(!(KMlistt$R1[k, j + 1] == 0)) {
                    KMlistt$X0[k, j + 2] <- (m*sum(ifelse(R == 1, R/sp *
                         I(Y == ttau[k]) * delta^(1-j) * (1-delta)^j *
                         I(X == 0), 0)))/ifelse(tracta.NA,
                         sum(R/sp * I(Y == ttau[k]) * delta^(1-j) *
                         (1-delta)^j), m)
                    KMlistt$X1[k, j + 2] <- m - KMlistt$X0[k, j + 2]
```

```
                }
            }
        }
        KMlistt$R1[K + 1, 2] <- sum(R * I(Y == tmax))
        KMlistt$R0[K + 1, 2] <- sum((1 - R) * I(Y == tmax))
        m <- KMlistt$R1[K + 1, 2] + KMlistt$R0[K + 1, 2] * as.integer(tracta.NA)
        if(!(KMlistt$R1[K + 1, 2] == 0)) {
            KMlistt$X0[K + 1, 3] <- (m * sum(ifelse(R == 1, R/sp *
                I(Y == tmax) * I(X == 0), 0)))/ifelse(tracta.NA,
                sum(R/sp * I(Y == tmax)), m)
            KMlistt$X1[K + 1, 3] <- m - KMlistt$X0[K + 1, 3]
        }
        KMlistt$nef[1] <- sum(KMlistt$X0[1:K, 2:3]) + KMlistt$X0[K + 1, 3]
        KMlistt$nef[2] <- sum(KMlistt$X1[1:K, 2:3]) + KMlistt$X1[K + 1, 3]
        KMlistt$nef[3] <- KMlistt$nef[1] + KMlistt$nef[2]
        KMlistt$X0[1, 1] <- KMlistt$nef[1]
        KMlistt$X0[1, 4] <- 1 - KMlistt$X0[1, 2]/KMlistt$nef[1]
        KMlistt$X1[1, 1] <- KMlistt$nef[2]
        KMlistt$X1[1, 4] <- 1 - KMlistt$X1[1, 2]/KMlistt$nef[2]
        for(k in 2:K) {
            KMlistt$X0[k, 1] <- KMlistt$X0[k-1,1] - sum(KMlistt$X0[k-1,2:3])
            KMlistt$X0[k, 4] <- ifelse(!(KMlistt$X0[k, 1] == 0),
                KMlistt$X0[k - 1, 4] * (1 - KMlistt$X0[k,2]/KMlistt$X0[k,1]),
                KMlistt$X0[k - 1, 4])
            KMlistt$X1[k, 1] <- KMlistt$X1[k-1,1] - sum(KMlistt$X1[k-1,2:3])
            KMlistt$X1[k, 4] <- ifelse(!(KMlistt$X1[k, 1] == 0),
                KMlistt$X1[k - 1, 4] * (1 - KMlistt$X1[k,2]/KMlistt$X1[k,1]),
                KMlistt$X1[k - 1, 4])
        }
        KMlist <<- KMlistt
        m <<- c(as.vector(t((KMlist$R1 + KMlist$R0)[1:K, 1:2])),
            KMlist$R1[K + 1, 2] + KMlist$R0[K + 1, 2])
}
```

Function `expit(x)`: Computation of the inverse of the logit function

```
"expit"<-function(x)
{
    exp(x)/(1 + exp(x))
}
```

Function `gendata(tmax, p, n, alpha, tau, beta, sigma)`: Generation of the data for the simulation, in a window $(0, T_{max}]$, with $P(X = 1) = p$, sample size n, non-response pattern with $\boldsymbol{\alpha}$ as ignorable parameter and τ as non-ignorable parameter and reference Weibull distributions $W(\beta, \sigma)$

```
"gendata"<-function(tmax = 3 * 365, p = 0.3, n = 100, alpha = c(-0.75,
    0.005, 1), tau = 0, beta = c(6.7, 1), sigma = c(1.4, -0.6))
{
    XX <<- rbinom(n, 1, p)
    TT <<- rweibull(n, 1/(sigma[1] + sigma[2] * XX), exp(beta[1] +
        beta[2] * XX))
    CC <<- runif(n, 0, tmax)
    YY <<- ifelse(CC < TT, CC, TT)
    Y <<- YY
    delta <<- ifelse(CC < TT, 0, 1)
    tsp <<- expit(alpha[1] + alpha[2] * Y + alpha[3] * delta + tau * XX)
    R <<- rbinom(n, 1, tsp)
    X <<- XX
    X[R == 0] <<- NA
}
```

Function `group(data, cuts, tmax)`: The function groups the data in `data` according to the partition defined by `cuts` and `tmax`

```
"group"<-function(data, cuts, tmax)
{
     for(i in 1:length(data)) {
          data[i] <- ifelse(data[i] <= max(cuts), cuts[sum(cuts <
               data[i]) + 1], tmax)
     }
     data
}
```

Function `ind(Y, delta, cuts, tmax)`: Returns the index in $\mathcal{J}$ of an observation $(Y_i, \delta_i)'$ according to the partition defined by `cuts` and `tmax`

```
"ind"<-function(Y, delta, cuts, tmax)
{
     pre <- sum(cuts < Y)
     ifelse(pre < K, ifelse(delta == 1, 2*pre+1, 2*pre+2), 2*K+1)
}
```

Function `norma(vector)`: Returns the norm of `vector`

```
"norma"<-function(vector)
{
     value <- sqrt(sum(vector^2))
     value
}
```

Function `phi2(r, Y, delta, X, R)`: Computes the function $\phi^{(2)}(.) = (1, Y, Y^2)'$ (or $\phi^{(2)}(.) = 1$ if we are in the MCAR case)

```
"phi2"<-function(r = 0, Y, delta, X, R)
{
     if(dimq == 1)
          value <- rep(1, length(Y))
     else value <- c(1, Y, Y^2)
     value
}
```

Function `sa(xop, opgrid, t, mcaranal, first, last, step)`: Returns a sensitivity analysis vector/matrix `s` for the covariate selected by `xop` and grid specified by `opgrid`, at time `t`. The function performs the MCAR analysis and a range of NI analysis

```
"sa" <- function(xop = 1, opgrid = 2, t = 52 * 7, mcaranal = FALSE,
     first = -6, last = 6, step = 0.1)
{
     library(Matrix)
     YY <<- SV
     Y <<- YY
     delta <<- deltaTB
     if(xop == 1) {
          X <<- C4
          R <<- RC4
     }
     else {
          X <<- PP
          R <<- RPP
     }
```

```
    ttau <<- switch(opgrid,
        (1:36) * 30,
        (1:154) * 7)
    K <<- length(ttau)
    tmax <<- 1085
    n <<- 494
    tint <- sum(ttau <= t)
    mcaranal <<- mcaranal
    if(mcaranal) {
        dimq <<- 1
        scalpha(0, 0, 0, 0)
        EGKM()
        s <<- c(KMlist$X0[tint, 4], se(0)[tint], KMlist$X1[tint, 4],
            se(1)[tint])
    }
    else {
        s <<- matrix(rep(0, 4 * ((last - first)/step + 1)), ncol = 4)
        dimq <<- 3
        i <- 1
        for(tau in seq(first, last, step)) {
            print(tau)
            scalpha(0, 0, 0, tau)
            EGKM()
            s[i, ] <<- c(KMlist$X0[tint, 4], se(0)[tint],
                KMlist$X1[tint, 4], se(1)[tint])
            i <- i + 1
            print(tau)
        }
    }
    s
}
```

Function `scalpha(alpha0, alpha1, alpha2, tau)`: Computes the estimation of the parameter $\boldsymbol{\alpha}$ in a non-response model with non-ignorable parameter τ

```
"scalpha"<-function(alpha0 = 0, alpha1 = 0, alpha2 = 0, tau = 0)
{
     Y <- YY
     if(mcaranal) {
          sp <- sum(R * phi2(, Y, delta))/sum(phi2(, Y, delta))
          sp <<- rep(sp, length(Y))
          alphan <- NA
     }
     else {
          count <- 0
          alphao <- c(alpha0, alpha1, alpha2)
          repeat {
               repeat {
                    slogitp <- ifelse(R == 1, alphao[1] + alphao[2] * Y
                         + alphao[3] * delta + tau * X, 0)
                    J <- matrix(rep(0, 9), ncol = 3)
                    for(i in 1:n) {
                         J <- J - R[i] * exp( - slogitp[i]) * matrix(c(1,
                              Y[i], Y[i]^2, Y[i], Y[i]^2, Y[i]^3,
                              delta[i], Y[i] * delta[i], Y[i]^2 * delta[i]),
                              ncol = 3)
                    }
                    if(!(prod(eigen(J)$values) == 0))
                         break
                    alphao <- alphao + c(0.01, -0.01, 0.01)
               }
               fo <- c(sum(R * exp( - slogitp)) - sum((1 - R)),
                    sum(R * exp( - slogitp) * Y) - sum((1 - R) * Y),
                    sum(R * exp( - slogitp) * Y^2) - sum((1 - R) * Y^2))
               alphan <- as.vector(alphao - ginverse(J) %*% fo)
```

```
                count <- count + 1
                dif <- norma(alphan - alphao)
                if((dif/norma(alphao) < 1e-010) || (count == 100))
                    break
                alphao <- alphan
            }
            sp <<- expit(ifelse(R == 1, alphan[1] + alphan[2] * Y +
                alphan[3] * delta + tau * X, 0))
        }
        alphan
}
```

Function `se(x)`: Returns the vector of standard errors for the EGKM estimator at each time of the partition, for the category `x`

```
"se"<-function(x = 0)
{
    M <- cSTG(x)
    sqrt(diag(M %*% (cLam1(x) + cLam2(x)) %*% t(M))/n)
}
```

Function `Ui(i, x)`: Computes the contribution of the *i*-th individual of the sample to the estimating equations for the category `x`

```
"Ui"<-function(i = 1, x = 0)
{
     Y <- Y[i]
     delta <- delta[i]
     X <- X[i]
     R <- R[i]
     j <- ind(Y, delta, ttau, tmax)
     beta <- beta(x)
     Ui1 <- rep(0, 2 * K + 1)
     if(R == 1)
          Ui1[j] <- (R * (m[j] * I(X == x) - beta[j]))/sp[i]
     Ui2 <- ((1 - R) - (R * (1 - sp[i]))/sp[i]) * phi2(0,Y,delta,X,R)
     Ui <- c(Ui1, Ui2)
     Ui
}
```