Supplementary Material

R Codes for Key Steps of the Case Study

```
library(reslife)

# 1. Predict residual survival time for censored cases
#####    T_D: TIME OF DEATH
## data: covarites, MON_TO_DEATH, DEATH_FLAG (one row per patient)

model <- flexsurvreg(Surv(MON_TO_DEATH,DEATH_FLAG)
                ~ NCIINDEX+URBRUR+GRADE+AGE+RACE+SEX+STAGE+SURG_CHEM,
                    data=data, dist="gengamma")
data$T_D = rep(NA, length(data$PATIENT_ID))

for (i in 1:length(data$PATIENT_ID)){

  if (data$DEATH_FLAG[i] == 1){
    data$T_D[i]<- data$MON_TO_DEATH[i]-1
  }
  if (data$DEATH_FLAG[i] == 0){
    df_new = data.frame(NCIINDEX = data$NCIINDEX[i],
                        URBRUR = data$URBRUR[i],
                        GRADE = data$GRADE[i],
                        AGE = data$AGE[i],
                        RACE = data$RACE[i],
                        SEX = data$SEX[i],
                        STAGE = data$STAGE[i],
                        SURG_CHEM = data$SURG_CHEM[i])

    data$T_D[i] <- data$MON_TO_DEATH[i]+ reslifefsr(obj = model,
                    life = data$MON_TO_DEATH[i], p = .5, type = ’median’,
                    newdata = df_new) - 1
  }
}

# 2. Create Design Matrix for a given set of Tau
# data.ext: covarites, T_D, DEATH_FLAG, MONTH, TOT_CHRG (a+b+1 rows per patient)
no_of_parameters = 4
design_mat_para = matrix(data = NA, nrow = nrow(data.ext),
                         ncol = (9*no_of_parameters))
pat_id = unique(data.ext$PATIENT_ID)

# Counting variable for cumulative per patient data points
pat_id=unique(data.ext$PATIENT_ID)
data_per_pat_count=c(NA)
for (i in 1:length(pat_id)){
```

```
  data_per_pat_count[i]=nrow(subset(data.ext,PATIENT_ID==pat_id[i]))
}
cumsum_data_len=c(0,cumsum(data_per_pat_count))

#per person
for (i in 1:length(pat_id)) {

  input_pat_data=subset(data.ext,PATIENT_ID==pat_id[i])
  rownames(input_pat_data)=NULL
  #Intercept + Covariates
  X=c(1,as.numeric(input_pat_data[1,c("NCIINDEX", "URBRUR",
         "GRADE","AGE", "RACE","SEX","STAGE","SURG_CHEM")]))

  #Define positive function (X>Y)
  posit_func=function(x,y){val=ifelse(x-y>0,x-y,0);  return(val)}

  #Design mat
  D_mat_para=matrix(data=NA,nrow = nrow(input_pat_data), ncol=(no_of_parameters))
  D_mat_para[,1]=1
  D_mat_para[,2]=ifelse(input_pat_data$MONTH<0,
                  posit_func(input_pat_data$MONTH,tau[1]), -tau[1])
  D_mat_para[,3]=ifelse(input_pat_data$MONTH>=0, - tau[2] +
                  posit_func(tau[2],input_pat_data$MONTH), 0)
  D_mat_para[,4]= posit_func(input_pat_data$MONTH,(input_pat_data$T_D - tau[3]))

  #The entire  sample Design Mat
  design_mat_para[c((cumsum_data_len[i]+1): cumsum_data_len[i+1]),]
                          =kronecker(t(X),D_mat_para)
}

# 3. Select the optimal lambda from a set of lambda, given the design matrix
data.pgee=data.frame(cbind(TOT_CHRG=data.ext$TOT_CHRG,
                           PATIENT_ID=data.ext$PATIENT_ID,
                           data.frame(design_mat_para)))
formula="TOT_CHRG ~ X1+X2+X3+X4+X5+X6+X7+X8+X9+X10+
                    X11+X12+X13+X14+X15+X16+X17+X18+X19+
                    X20+X21+X22+X23+X24+X25+X26+X27+X28+
                    X29+X30+X31+X32+X33+X34+X35+X36 - 1"
family <- gaussian(link = "identity")
lambda.vec <- exp(seq(-50,50,1))
G2_para_BIC = rbind(lambda.vec, rep(NA,length(lambda.vec)))

# Select optimal Lambda
for (i in 1:length(G2_para_BIC)) {
  G2_para_mod= PGEE(formula = formula, id = PATIENT_ID,
                    data = data.pgee, family=family,
```

```
                    lambda = lambda.vec[i], corstr = "independence")
  p <- length(G2_para_mod$coefficients!=0)
  n=G2_para_mod$nobs
  G2_para_BIC[2,i]= sum((G2_para_mod$residuals)^2) + p*log(n)
}

#Lambda selection
LAMMX=which(G2_para_BIC[2,] == min(G2_para_BIC[2,]), arr.ind = TRUE)
lam_opt=G2_para_BIC[1, min(LAMMX)]   #optimal

# 4. Calculate the coefficients based on the selected lambda
G2_para= PGEE(formula = formula, id = PATIENT_ID,
              data = data.pgee, family=family, lambda = lam_opt,
              corstr = "independence")
summary(G2_para)

# 5.    Repeat 2-3 based on for different combinations of taus.
# 6.    Select optimal Tau based on minimum RMSE.
```