20200330

Andrew Huang

3/30/2020

```
########## NEW gweight 20200218 ###########



gweight <- function(data,time,status){

  ##########
  n <- dim(data)[1]
  status[n] <- 1
  data01 <- data.frame(time,status)

  ##########
  KM.est<-KMestimator(time, status)
  data02 <- KM.est[[2]]
  rn <- dim(data02)[1]
  weight <- rep(0,rn)
  KM01 <- c(1,data02$KM)

  ##########
  for(i in 1:rn){
    weight[i] <- (KM01[i] - KM01[i+1])*n
  }

  ##########
  data02_1 <- data.frame(data02[,c('UZ','D','KM')],weight)
  n21 <- dim(data02_1)[1]
  rdata <- data.frame()

  ##########
  for(i in 1:n21){

    if(data02_1$D[i] > 1){
      w1 <- rep(data02_1$weight[i]/data02_1$D[i],data02_1$D[i])
      time <- rep(data02_1$UZ[i],data02_1$D[i])
      status <- rep(1,data02_1$D[i])
      td <- data.frame(time,status,w1)
    } else {
      w1 <- data02_1$weight[i]
      time <- data02_1$UZ[i]
      status <- 1
      td <- data.frame(time,status,w1)
    }
```

```r
    rdata <- rbind(rdata,td)
  }

  ##########
  data03_1 <- data.frame(data01[data01$status == 1,],w1 = rdata$w1)
  data03_2 <- data.frame(data01[data01$status == 0,],w1 = 0)
  data03 <- rbind(data03_1,data03_2) %>% arrange(time,-status)
  data_r <- data.frame(data,w1=data03$w1)



  return(data_r)
}

###########

get_synthetic <- function(data,time,status,w1){
  data_01 <- data %>% arrange(time)
  n_1 <- dim(data_01)[1]

  w2 <- w1
  data_01 <- cbind(data_01,w2)

  if(data_01[n_1,"status"] == 0){data_01[n_1,"w2"] <- 0}

  synthe <- numeric(n_1)

  for(j in 2:n_1){
    synthe[j] <- (data_01$time[j]-data_01$time[j-1])*data_01$w2[j]
  }

  data_01 <- cbind(data_01,synthe)

  Ag1 <- numeric(n_1)
  Ag1[1] <- data_01$time[1]

  for(j in 2:n_1){
    Ag1[j] <- Ag1[1] + sum(synthe[2:j])
  }

  data_01 <- cbind(data_01,Ag1)

  return(data_01)
}
```

```r
data_nlminb_cau <- read.csv("dataD-nlminb-cau.csv")

data_nlminb_t12 <- read.csv("dataD-nlminb-t12.csv")

data_nlminb_bst <- read.csv("dataD-nlminb-bst.csv")
```

```r
########## Obtain Kaplan-Meier estimate with the self-written functions;
KMestimator <- function (Z,delta)
{ UZ<-unique(Z)      #Unique observed times;
N<-length(UZ)
UZ.order<-order(UZ)
UZ<-UZ[UZ.order]             #sort data;
KM <- rep(0,N)
Y<-rep(0,N)
D<-rep(0,N)
D[1]<-sum(Z[delta==1]==UZ[1])
Y[1]<-sum(Z >= UZ[1])
KM[1] <- 1-D[1]/Y[1]            #this is for right continuous value
for (i in 2: N){
  D[i]<-sum(Z[delta==1]==UZ[i])
  Y[i]<-sum(Z >= UZ[i])
  KM[i] <- KM[i-1]*(1-D[i]/Y[i])
}

# Calculate variance and sdandard error;
sigma2.s<-rep(0,N)
for (i in 1: N){
  if (D[i]==Y[i]){
    sigma2.s[i]=0    #assign an arbitrary value, it does not affect the results;
  }
  else{
    sigma2.s[i]<-sum( (UZ[1:i]<=UZ[i])*(D[1:i]/(Y[1:i]*(Y[1:i]-D[1:i]))))
  }
  ## sigma2.s[i]<-sum( (UZ<=UZ[i])*(D/(Y*(Y-D))) ) #old version
  ##  Note the data is sorted by UZ;
  ##  Using this to avoid NaN for times smaller than the largest observation;
}
KM.var<-KM^2*sigma2.s
KM.se<-sqrt(KM.var)

Full<-data.frame(UZ,D,Y,KM,sigma2.s,KM.var,KM.se)
Reduced<-subset(Full, (Full$D>0))
list(Full, Reduced)
}


########## NewMethod using W_Prime

f_function_MD01_i <- function(beta_c,age=data_i$sage,age2=data_i$sage2,time=data_i$stime,w1=data_i$wp){
  beta0 <- beta_c[1]
  beta1 <- beta_c[2]
  beta2 <- beta_c[3]

  n <- length(age)

  f_function_MD01_i = sum( (log10(time) - (beta0*rep(1,n)+age*beta1+age2*beta2))^2 * w1 )
  return(f_function_MD01_i)
}
```

```r
########## Type I Discrete using W1

f_function_MD02_i <- function(beta_c,age=data_i$sage,age2=data_i$sage2,time=data_i$stime,w1=data_i$w1){
  beta0 <- beta_c[1]
  beta1 <- beta_c[2]
  beta2 <- beta_c[3]

  n <- length(age)

  f_function_MD02_i = sum((
    ifelse(log10(time) >=0,sqrt(log10(time)),0) -
      ifelse(beta0*rep(1,n)+age*beta1+age2*beta2>=0,sqrt(beta0*rep(1,n)+age*beta1+age2*beta2),0)
  )^2 * w1)
  return(f_function_MD02_i)
}


########## Least Square Method by Zhou

f_function_MD03_i <- function(beta_c,age=data_i$sage,age2=data_i$sage2,time=data_i$stime,w1=data_i$w1){
  beta0 <- beta_c[1]
  beta1 <- beta_c[2]
  beta2 <- beta_c[3]

  n <- length(age)

  f_function_MD03_i = sum( (log10(time) - (beta0*rep(1,n)+age*beta1+age2*beta2))^2 * w1 )
  return(f_function_MD03_i)
}


########## Least Absolute Deviation by Zhou

f_function_MD032_i <- function(beta_c,age=data_i$sage,age2=data_i$sage2,time=data_i$stime,w1=data_i$w1){
  beta0 <- beta_c[1]
  beta1 <- beta_c[2]
  beta2 <- beta_c[3]

  n <- length(age)

  f_function_MD032_i = sum( abs(log10(time) - (beta0*rep(1,n)+age*beta1+age2*beta2)) * w1 )
  return(f_function_MD032_i)
}
```

```r
get_mean_2 <- function(data = D_BFGS){
  intercept_mean_0 <- mean(data$ps_1v)
  age_mean_0 <- mean(data$ps_2v)
  age2_mean_0 <- mean(data$ps_3v)
  output0 <- c(intercept_mean_0,age_mean_0,age2_mean_0)
  names(output0) <- c("Intercept","Age","Age2")
  print(output0)

  intercept_mean_0_1 <- mean(data$ps1_1v)
```

```
  age_mean_0_1 <- mean(data$ps1_2v)
  age2_mean_0_1 <- mean(data$ps1_3v)
  output0_1 <- c(intercept_mean_0_1,age_mean_0_1,age2_mean_0_1)
  names(output0_1) <- c("Intercept","Age","Age2")
  print(output0_1)

  intercept_mean_1 <- mean(data$p1_1v)
  age_mean_1 <- mean(data$p1_2v)
  age2_mean_1 <- mean(data$p1_3v)
  output1 <- c(intercept_mean_1,age_mean_1,age2_mean_1)
  names(output1) <- c("Intercept","Age","Age2")
  print(output1)

    intercept_mean_0_2 <- mean(data$ps2_1v)
  age_mean_0_2 <- mean(data$ps2_2v)
  age2_mean_0_2 <- mean(data$ps2_3v)
  output0_2 <- c(intercept_mean_0_2,age_mean_0_2,age2_mean_0_2)
  names(output0_2) <- c("Intercept","Age","Age2")
  print(output0_2)

  intercept_mean_2 <- mean(data$p2_1v)
  age_mean_2 <- mean(data$p2_2v)
  age2_mean_2 <- mean(data$p2_3v)
  output2 <- c(intercept_mean_2,age_mean_2,age2_mean_2)
  names(output2) <- c("Intercept","Age","Age2")
  print(output2)

  intercept_mean_3 <- mean(data$p3_1v)
  age_mean_3 <- mean(data$p3_2v)
  age2_mean_3 <- mean(data$p3_3v)
  output3 <- c(intercept_mean_3,age_mean_3,age2_mean_3)
  names(output3) <- c("Intercept","Age","Age2")
  print(output3)

  intercept_mean_32 <- mean(data$p32_1v)
  age_mean_32 <- mean(data$p32_2v)
  age2_mean_32 <- mean(data$p32_3v)
  output32 <- c(intercept_mean_32,age_mean_32,age2_mean_32)
  names(output32) <- c("Intercept","Age","Age2")
  print(output32)

  age_mean_bj <- mean(data$pbj_2v, na.rm = TRUE)
  age2_mean_bj <- mean(data$pbj_3v, na.rm = TRUE)
  outputbj <- c(age_mean_bj,age2_mean_bj)
  names(outputbj) <- c("Age","Age2")
  print(outputbj)
}
```

```r
get_mean_2(data_nlminb_cau)
```

```
##   Intercept          Age         Age2 
##  1.749015361  0.121562451 -0.001250336 
##   Intercept          Age         Age2 
##  1.942068387  0.136204997 -0.001413944 
##   Intercept          Age         Age2 
##  1.942068387  0.136204997 -0.001413944 
##   Intercept          Age         Age2 
##  1.942068387  0.136204997 -0.001413944 
##   Intercept          Age         Age2 
##  3.733440085  0.123300360 -0.001240033 
##   Intercept          Age         Age2 
##  3.733440085  0.123300360 -0.001240033 
##   Intercept          Age         Age2 
##  3.733440085  0.123300360 -0.001240033 
##  Age Age2 
##  NaN  NaN
```

```r
get_mean_2(data_nlminb_t12)
```

```
##   Intercept         Age        Age2 
##  1.15761126  0.15577975 -0.00166238 
##    Intercept          Age         Age2 
##  0.945277310  0.161794439 -0.001725551 
##   Intercept         Age        Age2 
##  0.87630774  0.17330705 -0.00185787 
##    Intercept          Age         Age2 
## -1.577475177  0.420606338 -0.007273858 
##    Intercept          Age         Age2 
##  1.360822994  0.166337599 -0.001791161 
##    Intercept          Age         Age2 
##  1.492757496  0.163024441 -0.001754695 
##    Intercept          Age         Age2 
##  1.692165882  0.153690595 -0.001669929 
##          Age         Age2 
##  0.148073586 -0.001575568
```

```r
get_mean_2(data_nlminb_bst)
```

```
##    Intercept          Age         Age2
##  1.090793908  0.158758751 -0.001694361
##    Intercept          Age         Age2
##  1.039141941  0.159932470 -0.001706469
##    Intercept          Age         Age2
##  1.021523179  0.170999167 -0.001846305
##    Intercept          Age         Age2
## -2.142384980  0.481257695 -0.008541101
##    Intercept          Age         Age2
##  1.145139072  0.168515192 -0.001821011
##    Intercept          Age         Age2
##  1.182592934  0.167753368 -0.001812857
##    Intercept          Age         Age2
##  1.537061850  0.155450843 -0.001653731
##          Age         Age2
##  0.148403379 -0.001576902
```