Supporting information for "On the integration of decision trees with mixture cure model"

Wisdom Aselisewine and Suvra Pal

Department of Mathematics, University of Texas at Arlington,
Arlington, Texas 76019, United States

A1. R code for data generation in Section 4

```
mydata = function(n,alpha,beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9,
beta10,cens,setting){
  z1 = rbern(n,prob = 0.5) # z1 to z4 are the four binary covariates generated
  z2 = rbern(n,prob = 0.3) # independently from Bernoulli distribution.
  z3 = rbern(n,prob = 0.5)
  z4 = rbern(n,prob = 0.7)
  z5 = rnorm(n,mean=0,sd=1) # z5 to z10 are the six continuous covariates generated
  z6 = rnorm(n,mean=0,sd=1) # independently from standard normal distributions.
  z7 = rnorm(n,mean=0,sd=1)
  z8 = rnorm(n,mean=0,sd=1)
  z9 = rnorm(n,mean=0,sd=1)
  z10 = rnorm(n,mean=0,sd=1)

  piz = rep(NA,n)  # this is the uncured probability pi(z)

  if(setting==1){ # non-linear classification boundary
    piz = (exp(0.3+(10*z1*z1*z2)-(5*z2*z2))/(1+exp(0.3+(10*z1*z1*z2)-(5*z2*z2))))
  }

  if(setting==2){ # complex classification boundary with high dimension
    piz = exp(-exp(-0.8*z1*z2+1.1*z2*z4+0.5*z3+0.2*z7*z7-1.3*sin(z5*z6)+
    1.9*cos(z7*z8)-1.5*exp(z5*z6*z7)-1.6*z7*z8*z9*z10+0.8*(z6*z7)*(z8*z9*z8*z9)
    +1.8*cos(z5*z6*z7*z8*z9)+1.2*(abs(z6*z7*z8*z9*z10)^0.5)-2.4))
  }

    C = runif(n,0,cens) # censoring time
  U = runif(n,0,1)
  Y = rep(NA,n) # observed lifetime
  D = rep(NA,n) # censoring indicator
  J = rep(NA,n) # cured indicator (J=0 implies cured)
```

```
  Sp = rep(NA,n) # overall (population) survival function
  S1 = rep(NA,n) # survival function of susceptible group
  for(i in 1:n){
    if(U[i]<= 1-piz[i]){
      Y[i] = C[i]
      D[i] = 0
      J[i] = 0
      S1[i] = exp(-((Y[i]/exp(-((beta1*z1[i])+(beta2*z2[i])+(beta3*z3[i])+
      (beta4*z4[i])+(beta5*z5[i])+(beta6*z6[i])+(beta7*z7[i])+(beta8*z8[i])+
      (beta9*z9[i])+(beta10*z10[i]))/alpha))^alpha))
      Sp[i] = (1-piz[i]) + (piz[i]*exp(-((Y[i]/exp(-((beta1*z1[i])+(beta2*z2[i])+
      (beta3*z3[i])+ (beta4*z4[i])+(beta5*z5[i])+(beta6*z6[i])+(beta7*z7[i])+
      (beta8*z8[i])+(beta9*z9[i])+(beta10*z10[i]))/alpha))^alpha)))
    }
    else{
      T =  rweibull(1,shape=alpha,scale=exp(-((beta1*z1[i])+(beta2*z2[i])+
      (beta3*z3[i])+(beta4*z4[i])+(beta5*z5[i])+(beta6*z6[i])+(beta7*z7[i])+
      (beta8*z8[i])+(beta9*z9[i])+(beta10*z10[i]))/alpha) )
      Y[i] = min(T,C[i])
      J[i] = 1
      S1[i] = exp(-((Y[i]/exp(-((beta1*z1[i])+(beta2*z2[i])+(beta3*z3[i])+
      (beta4*z4[i])+(beta5*z5[i])+(beta6*z6[i])+(beta7*z7[i])+(beta8*z8[i])+
      (beta9*z9[i])+(beta10*z10[i]))/alpha))^alpha))
      Sp[i] = (1-piz[i])+(piz[i]*exp(-((Y[i]/exp(-((beta1*z1[i])+(beta2*z2[i])
      +(beta3*z3[i])+(beta4*z4[i])+(beta5*z5[i])+(beta6*z6[i])+(beta7*z7[i])+
      (beta8*z8[i])+(beta9*z9[i])+(beta10*z10[i]))/alpha))^alpha)))
      if(Y[i]==C[i]){
        D[i] = 0
      }
      else{
        D[i] = 1
      }
    }
  }
  return(data.frame(Y,D,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,J,uncure=piz,S1,Sp=Sp))
} # function to generate data under 3 different classification boundaries
```

A2. R code for DT-based EM algorithm in Section 4

```
library(e1071)
library(survival)
library(rpart)

smsurv <-function(Time,Status,X,beta,w,model){
```

```
death_point <- sort(unique(subset(Time, Status==1)))
if(model=='ph') coxexp <- exp((beta)%*%t(X[,-1]))
lambda <- numeric()
event <- numeric()
for(i in 1: length(death_point)){
event[i] <- sum(Status*as.numeric(Time==death_point[i]))
if(model=='ph')  temp <- sum(as.numeric(Time>=death_point[i])*w*drop(coxexp))
if(model=='aft')  temp <- sum(as.numeric(Time>=death_point[i])*w)
temp1 <- event[i]
lambda[i] <- temp1/temp
}
HHazard <- numeric()
for(i in 1:length(Time)){
HHazard[i] <- sum(as.numeric(Time[i]>=death_point)*lambda)
if(Time[i]>max(death_point))HHazard[i] <- Inf
if(Time[i]<min(death_point))HHazard[i] <- 0
}
survival <- exp(-HHazard)
list(survival=survival)
}



em.dt.RC = function(Time,Status,X,Z,uncureprob,beta,s0,emmax,eps){

w <- Status
n <- length(Status)
s <- smsurv(Time,Status,X,beta,w,model="ph")$survival
convergence<- 1000;i <-1
while (convergence > eps & i < emmax){
#print(i)
survival<-drop(s^(exp((beta)%*%t(X[,-1]))))
# E step
w <- Status+(1-Status)*(uncureprob*survival)/((1-uncureprob)+uncureprob*survival)
# M step
multipleuncureprob=matrix(1:5*n, nrow=n,ncol=5)
for (j in 1:n){multipleuncureprob[j,] = rbinom(5, size = 1, prob = w[j])}

uncureprob1 = c(1,1)
uncureprob2 = c(1,1)
uncureprob3 = c(1,1)
uncureprob4 = c(1,1)
uncureprob5 = c(1,1)

for (j in 1:n){uncureprob1[j] = multipleuncureprob[j,1]}
```

```
for (j in 1:n){uncureprob2[j] = multipleuncureprob[j,2]}
for (j in 1:n){uncureprob3[j] = multipleuncureprob[j,3]}
for (j in 1:n){uncureprob4[j] = multipleuncureprob[j,4]}
for (j in 1:n){uncureprob5[j] = multipleuncureprob[j,5]}

for (j in 1:n){uncureprob1[j] = uncureprob1[j]}
for (j in 1:n){uncureprob2[j] = uncureprob2[j]}
for (j in 1:n){uncureprob3[j] = uncureprob3[j]}
for (j in 1:n){uncureprob4[j] = uncureprob4[j]}
for (j in 1:n){uncureprob5[j] = uncureprob5[j]}

uncureprob1 = as.factor(uncureprob1)
uncureprob2 = as.factor(uncureprob2)
uncureprob3 = as.factor(uncureprob3)
uncureprob4 = as.factor(uncureprob4)
uncureprob5 = as.factor(uncureprob5)

update_cureb = c(1,1)

obj3 <- tune.rpart(uncureprob1~Z[,-1], data = data, minsplit = seq(41,51,5),
cp = seq(0.002,0.012,0.005),xval = 3 )
bc<-obj3$best.parameters[1]
bg<-obj3$best.parameters[2]

mod1 <- rpart(formula = uncureprob1~Z[,-1], data = data,method = "class",
control = rpart.control(minsplit = bc[[1]], cp = bg[[1]]))
proba1 <- predict(mod1, newdata = data,type = "prob")
update_cureb1<-c(1,1)
for (z in 1:n){update_cureb1[z]<-proba1[z,colnames(proba1)==1]}
uncureprob1<-as.numeric(as.character(uncureprob1))

mod2 <- rpart(formula = uncureprob2~Z[,-1], data = data,method = "class",
control = rpart.control(minsplit = bc[[1]], cp = bg[[1]]))
proba2 <- predict(mod2, newdata = data,type = "prob")
update_cureb2<-c(1,1)
for (z in 1:n){update_cureb2[z]<-proba2[z,colnames(proba2)==1]}
uncureprob2<-as.numeric(as.character(uncureprob2))

mod3 <- rpart(formula = uncureprob3~Z[,-1], data = data,method = "class",
control = rpart.control(minsplit = bc[[1]], cp = bg[[1]]))
proba3 <- predict(mod3, newdata = data,type = "prob")
update_cureb3<-c(1,1)
for (z in 1:n){update_cureb3[z]<-proba3[z,colnames(proba3)==1]}
uncureprob3<-as.numeric(as.character(uncureprob3))

mod4 <- rpart(formula = uncureprob4~Z[,-1], data = data,method = "class",
```

```
control = rpart.control(minsplit = bc[[1]], cp = bg[[1]]))
proba4 <- predict(mod4, newdata = data,type = "prob")
update_cureb4<-c(1,1)
for (z in 1:n){update_cureb4[z]<-proba4[z,colnames(proba4)==1]}
uncureprob4<-as.numeric(as.character(uncureprob4))

mod5 <- rpart(formula = uncureprob5~Z[,-1], data = data,method = "class",
control = rpart.control(minsplit = bc[[1]], cp = bg[[1]]))
proba5 <- predict(mod5, newdata = data,type = "prob")
update_cureb5<-c(1,1)
for (z in 1:n){update_cureb5[z]<-proba5[z,colnames(proba5)==1]}
uncureprob5<-as.numeric(as.character(uncureprob5))

for (z in 1:n){update_cureb[z]<-(update_cureb1[z]+update_cureb2[z]
+update_cureb3[z]+update_cureb4[z]+update_cureb5[z])/5}
update_beta <- coxph(Surv(Time, Status)~X[,-1]+offset(log(w)),subset=w!=0,
method="breslow")$coef
update_s <-smsurv(Time,Status,X,beta,w,model="ph")$survival

convergence<-sum(c(mean(update_cureb)-mean(uncureprob),update_beta-beta,
mean(update_s)-mean(s))^2)
uncureprob <- update_cureb
beta <- update_beta
s<-update_s
i <- i+1
#print(i)
}
S1 = drop(s^(exp((beta)%*%t(X[,-1])))) # survival function of susceptible group
Sp = (1-uncureprob)+(uncureprob*S1)
em.dt <- list(latencyfit=beta,Uncureprob=uncureprob,S0=s,S1=S1,Sp=Sp,
tau=convergence, Mod1=mod1, Mod2=mod2, Mod3=mod3, Mod4=mod4, Mod5=mod5)
}
```

A3. Plots in Section 5

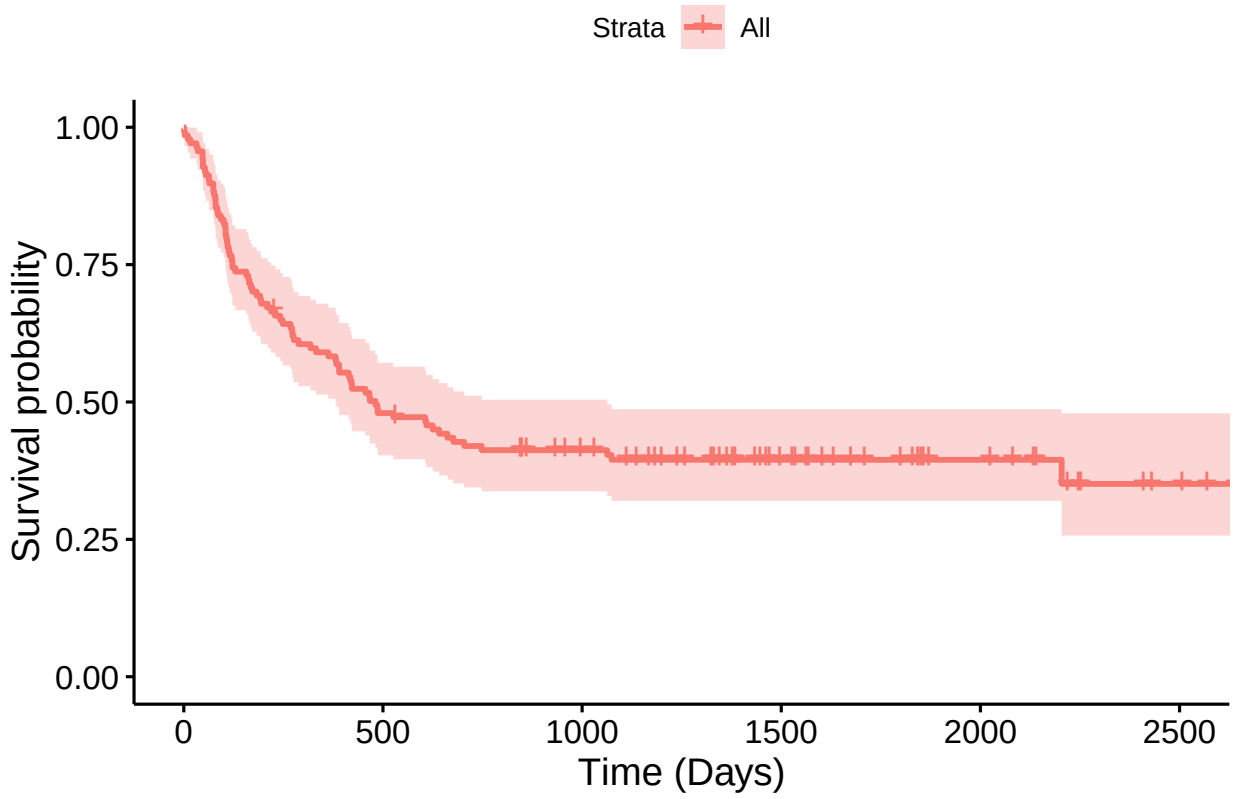

Figure A3.1: Kaplan-Meier survival curve for the leukemia data

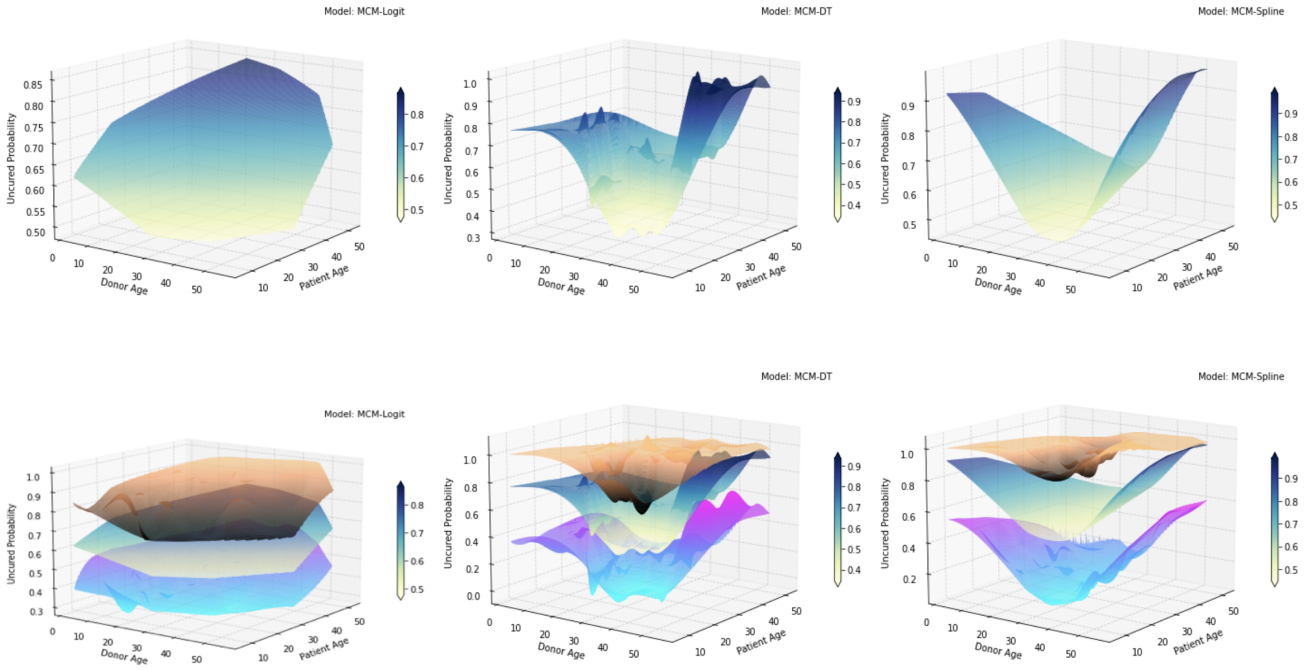

Figure A3.2: 3-dimensional surface plane of uncured probabilities, along with 95% confidence bounds, as a function of patient's age and donor's age