Supplemental material 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

```
mydata = function(n,alpha,beta1,beta2,cens,setting){
  z1 = rnorm(n,mean=0,sd=1) # z1 and z2 are the two continuous covariates
                # generated independently from standard normal distributions
  z2 = rnorm(n,mean=0,sd=1)

  piz = rep(NA,n) # this is the uncured probability pi(z)
  if(setting==1){ # linear classification boundary
    piz = (exp(0.3-(5*z1)-(3*z2))/(1+exp(0.3-(5*z1)-(3*z2))))
  }
  if(setting==2){ # non-linear classification boundary
    piz =  (exp(0.3-(5*z1*z2)-(3*z1*z2))/(1+exp(0.3-(5*z1*z2)-(3*z1*z2))))
  }

  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]))/alpha))^alpha))
      Sp[i] = (1-piz[i]) + (piz[i]*exp(-((Y[i]/exp(-((beta1*z1[i])
              +(beta2*z2[i]))/alpha))^alpha)))
    }
    else{
```

```
      T =  rweibull(1,shape=alpha,scale= exp(-((beta1*z1[i])+(beta2*z2[i])))/alpha)
      Y[i] = min(T,C[i])
      J[i] = 1
      S1[i] = exp(-((Y[i]/exp(-((beta1*z1[i])+(beta2*z2[i])))/alpha))^alpha))
      Sp[i] = (1-piz[i]) + (piz[i]*exp(-((Y[i]/exp(-((beta1*z1[i])
              +(beta2*z2[i])))/alpha))^alpha)))
      if(Y[i]==C[i]){
        D[i] = 0
      }
      else{
        D[i] = 1
      }
    }
  }
  return(data.frame(Y,D,z1,z2,J,uncure=piz,S1,Sp=Sp))
} # function to generate data under 3 different classification boundaries
```

A2. R code for DT-based EM algorithm

```
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 code

em.dt.RC = function(Time, Status, X, X1, Z, Z1, offsetvar, uncureprob, uncurepred, beta,
emmax, eps, data, testdata)
{

  w <- Status
  n <- length(Status)
  m <- dim(testdata)[1]
  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)
update_pred<-c(1,1)

obj3 <- tune.rpart(uncureprob1~Z[,-1], data = data, minsplit =c(11,20,25),
cp = c(0.001,0.005,0.01))
bc<-obj3$best.parameters[1]
bg<-obj3$best.parameters[2]

mod1 <- rpart(formula = uncureprob1~z1+z2, data = data,
method = "class", control = rpart.control(minsplit = bc[[1]],
minbucket = round(bc[[1]]/3), cp = bg[[1]]),xval = 10,
parms = list(split="gini"))

cp.min1<-mod1$cptable[which.min(mod1$cptable[,"xerror"]),"CP"]
```

```
tree1<-prune(mod1, cp=cp.min1)

proba1 <- predict(tree1, newdata = data,type = "prob")
cproba1 <- predict(tree1, newdata = testdata,type = "prob")

update_cureb1<-c(1,1)
update_pred1<-c(1,1)
for (z in 1:n){update_cureb1[z]<-proba1[z,colnames(proba1)==1]}
for (d in 1:m){update_pred1[d]<-cproba1[d,colnames(cproba1)==1]}
uncureprob1<-as.numeric(as.character(uncureprob1))

mod2 <- rpart(formula = uncureprob2~z1+z2, data = data,
method = "class",control = rpart.control(minsplit = bc[[1]],
minbucket = round(bc[[1]]/3), cp = bg[[1]]),xval = 10,
parms = list(split="gini"))

cp.min2<-mod2$cptable[which.min(mod2$cptable[,"xerror"]),"CP"]
tree2<-prune(mod2, cp=cp.min2)

proba2 <- predict(tree2, newdata = data,type = "prob")
cproba2 <- predict(tree2, newdata = testdata,type = "prob")

update_cureb2<-c(1,1)
update_pred2<-c(1,1)
for (z in 1:n){update_cureb2[z]<-proba2[z,colnames(proba2)==1]}
for (d in 1:m){update_pred2[d]<-cproba2[d,colnames(cproba2)==1]}
uncureprob2<-as.numeric(as.character(uncureprob2))

mod3 <- rpart(formula = uncureprob3~z1+z2, data = data,
method = "class",control = rpart.control(minsplit = bc[[1]],
minbucket = round(bc[[1]]/3), cp = bg[[1]]),xval = 10,
parms = list(split="gini"))

cp.min3<-mod3$cptable[which.min(mod3$cptable[,"xerror"]),"CP"]
tree3<-prune(mod3, cp=cp.min3)

proba3 <- predict(tree3, newdata = data,type = "prob")
cproba3 <- predict(tree3, newdata = testdata,type = "prob")

update_cureb3<-c(1,1)
update_pred3<-c(1,1)
for (z in 1:n){update_cureb3[z]<-proba3[z,colnames(proba3)==1]}
for (d in 1:m){update_pred3[d]<-cproba3[d,colnames(cproba3)==1]}
uncureprob3<-as.numeric(as.character(uncureprob3))

mod4 <- rpart(formula = uncureprob4~z1+z2, data = data,
```

```
method = "class",control = rpart.control(minsplit = bc[[1]],
minbucket = round(bc[[1]]/3), cp = bg[[1]]),xval = 10,
parms = list(split="gini"))
cp.min4<-mod4$cptable[which.min(mod4$cptable[,"xerror"]),"CP"]
tree4<-prune(mod4, cp=cp.min4)

proba4 <- predict(tree4, newdata = data,type = "prob")
cproba4 <- predict(tree4, newdata = testdata,type = "prob")

update_cureb4<-c(1,1)
update_pred4<-c(1,1)
for (z in 1:n){update_cureb4[z]<-proba4[z,colnames(proba4)==1]}
for (d in 1:m){update_pred4[d]<-cproba4[d,colnames(cproba4)==1]}
uncureprob4<-as.numeric(as.character(uncureprob4))

mod5 <- rpart(formula = uncureprob5~z1+z2, data = data,
method = "class",control = rpart.control(minsplit = bc[[1]],
minbucket = round(bc[[1]]/3), cp = bg[[1]]),xval = 10,
parms = list(split="gini"))

cp.min5<-mod5$cptable[which.min(mod5$cptable[,"xerror"]),"CP"]
tree5<-prune(mod5, cp=cp.min5)

proba5 <- predict(tree5, newdata = data,type = "prob")
cproba5 <- predict(tree5, newdata = testdata,type = "prob")

update_cureb5<-c(1,1)
update_pred5<-c(1,1)
for (z in 1:n){update_cureb5[z]<-proba5[z,colnames(proba5)==1]}
for (d in 1:m){update_pred5[d]<-cproba5[d,colnames(cproba5)==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}
for (d in 1:m){update_pred[d]<-(update_pred1[d]+update_pred2[d]
             +update_pred3[d]+update_pred4[d]+update_pred5[d])/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
uncurepred <- update_pred
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,Uncurepred=uncurepred,S0=s,
  S1=S1,Sp=Sp,tau=convergence, Mod1=mod1, Mod2=mod2, Mod3=mod3, Mod4=mod4,
  Mod5=mod5)
}
```

A3. Additional figures

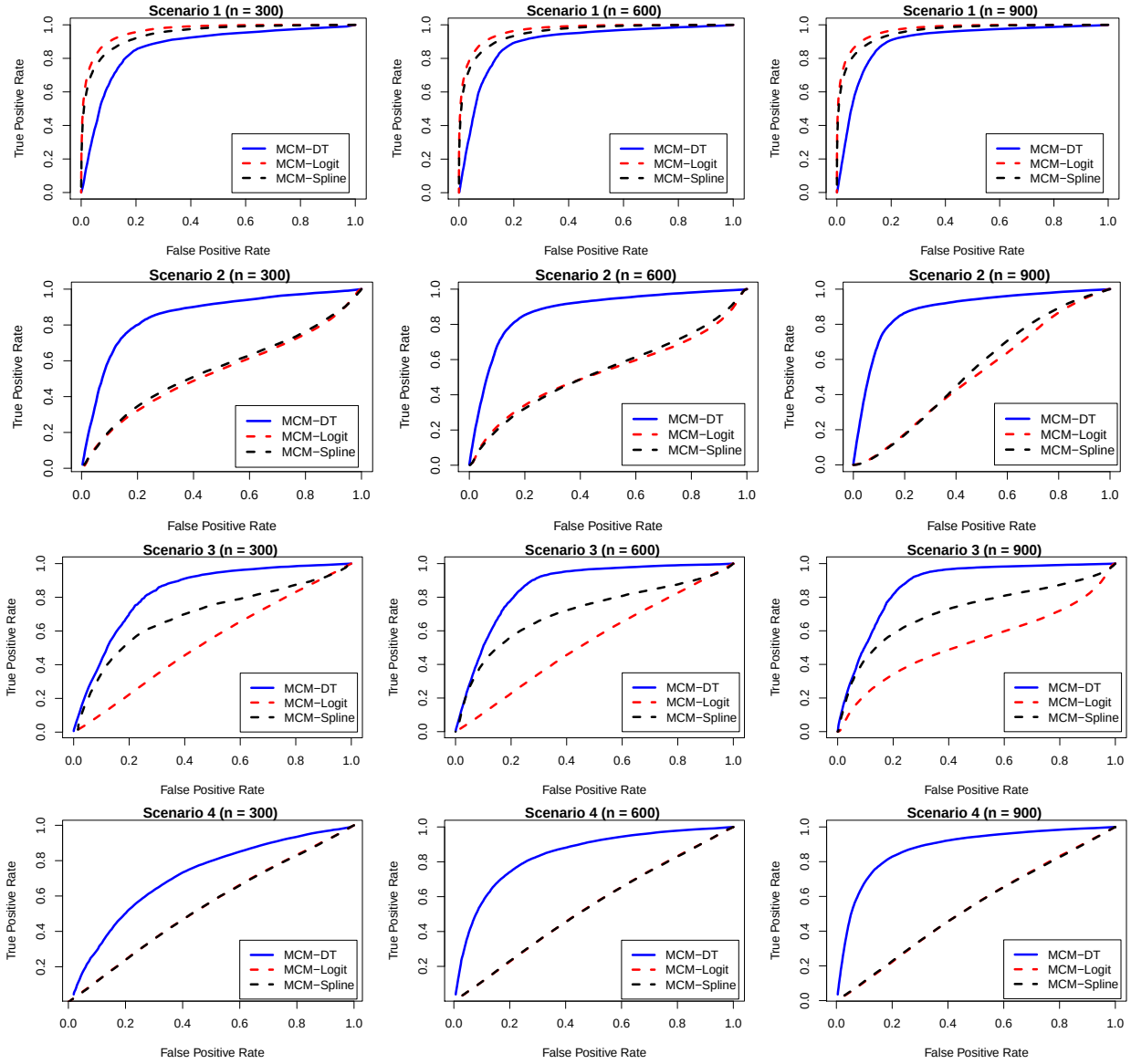

Figure A3.1: Averaged ROC curves for different models and sample sizes.

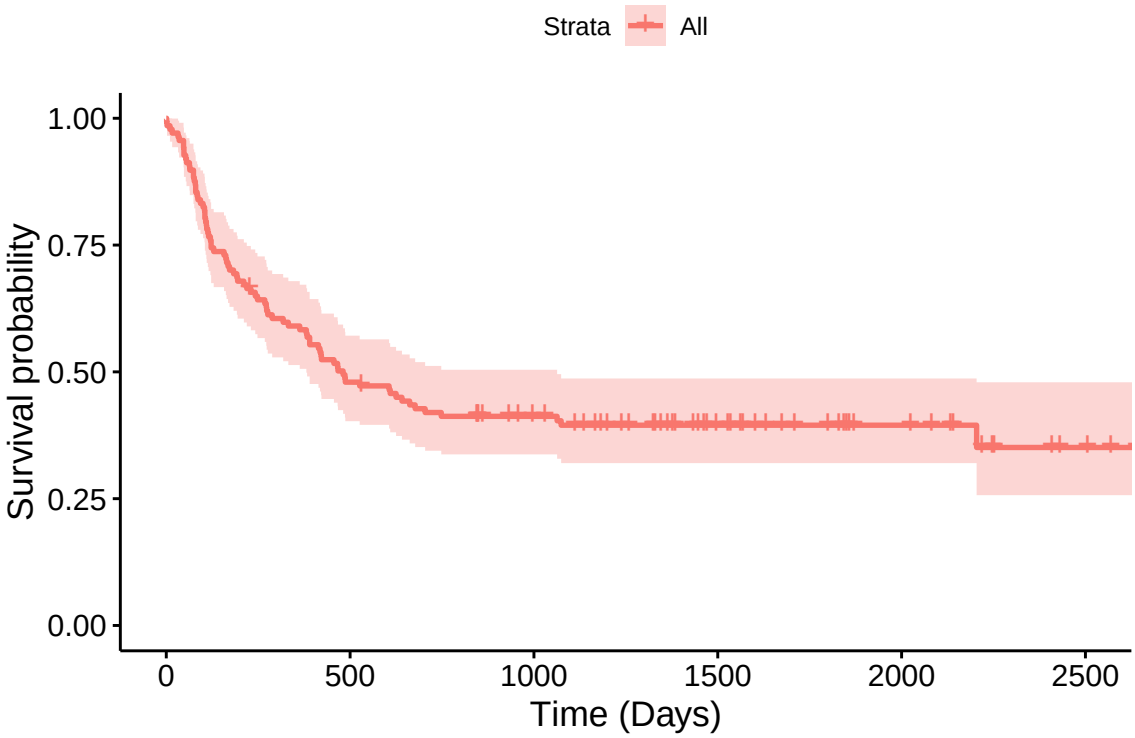

Figure A3.2: Kaplan-Meier survival curve for the leukemia data

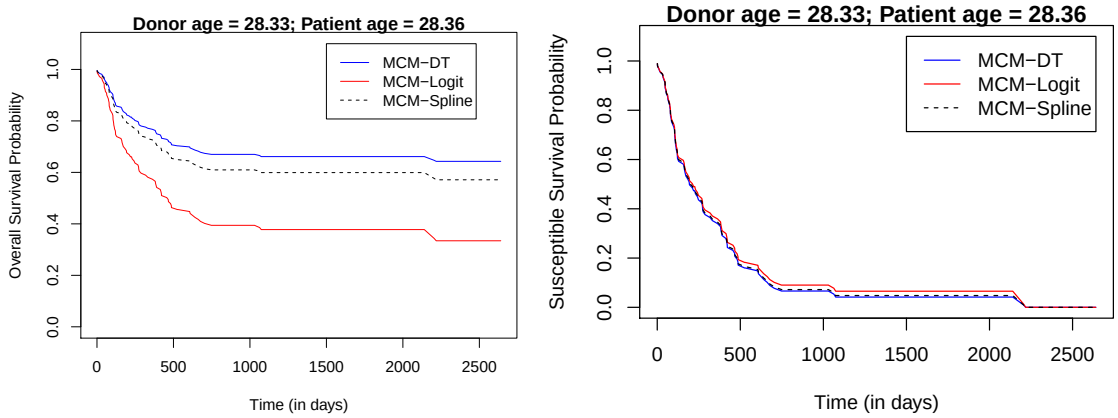

Figure A3.3: Predicted overall and susceptible survival probabilities for the mean values of donor's age and patient's based on the leukemia data

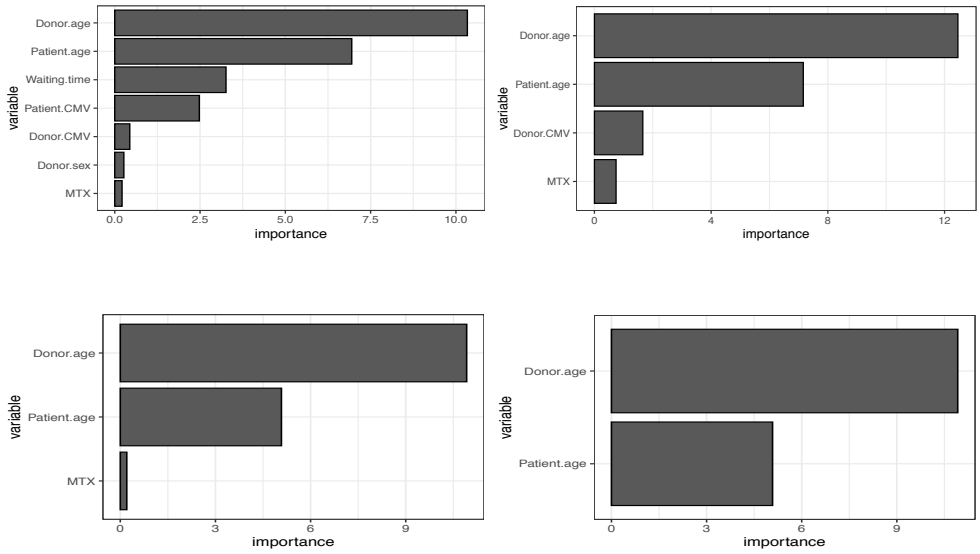

Figure A3.4: Relative variable importance plots corresponding to the leukemia data

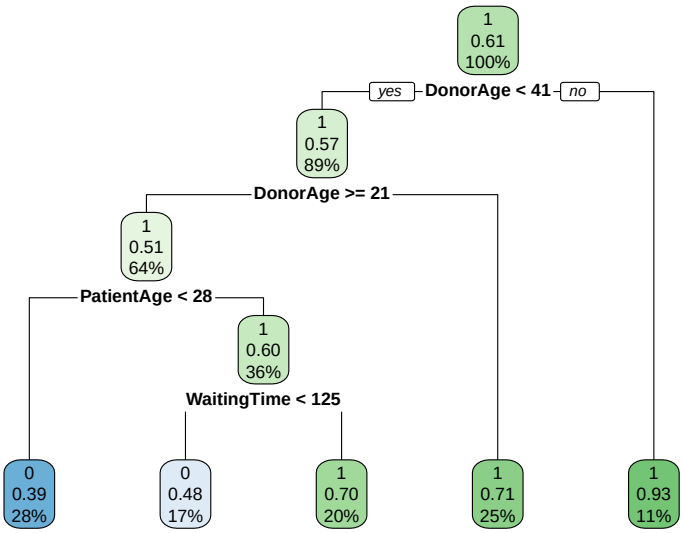

Figure A3.5: Decision tree plot corresponding to the leukemia data. Note that 0 represents cured group and 1 represents uncured group

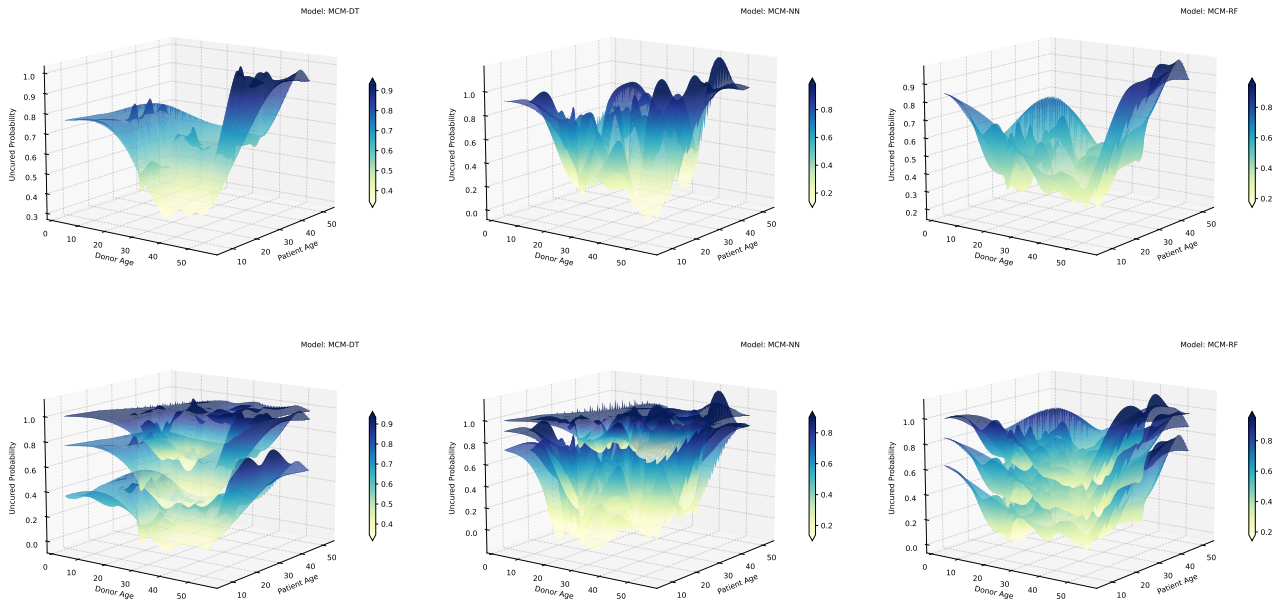

Figure A3.6: 3-dimensional surface plane of uncured probabilities, along with 95% confidence bounds, as a function of patient's age and donor's age for the DT, NN and RF models

A4. Additional tables

Table A4.1: Bias and MSE of the baseline survival probability for different models and $n = 600$

Scenario	Baseline Survival Probability ($S_{u0}(t)$)					
	Bias			MSE		
	DT	Spline	Logit	DT	Spline	Logit
1	0.0319	0.0338	0.0371	0.0017	0.0019	0.0023
2	0.0314	0.0269	0.0280	0.0015	0.0012	0.0013
3	0.0424	0.0396	0.0327	0.0027	0.0024	0.0017
4	0.0163	0.0164	0.0165	0.0009	0.0009	0.0009

Table A4.2: Estimation results corresponding to the latency parameters for $n = 900$

Scenario	Parameter	Estimate			Standard Error		
		Logit	Spline	DT	Logit	Spline	DT
1	β_1	0.798	0.666	0.423	0.116	0.124	0.286
	β_2	0.342	0.326	0.166	0.108	0.108	0.209
2	β_1	1.288	1.288	0.922	0.135	0.130	0.089
	β_2	0.877	0.891	0.607	0.103	0.103	0.091
3	β_1	1.139	0.892	0.857	0.137	0.118	0.102
	β_2	-0.104	0.261	0.316	0.180	0.114	0.104
	β_1	1.011	1.009	1.001	0.145	0.139	0.136
	β_2	0.888	0.885	0.891	0.135	0.153	0.144
4	β_3	0.525	0.524	0.521	0.082	0.071	0.071
	β_4	1.021	1.020	1.028	0.091	0.078	0.080
	β_5	-0.675	-0.673	-0.667	0.076	0.074	0.055

Table A4.3: Comparison of DT, NN, and RF models through latency parameter estimation for $n = 900$

Scenario	Parameter	Estimate			Standard Error		
		DT	NN	RF	DT	NN	RF
1	β_1	0.423	0.414	0.588	0.286	0.289	0.133
	β_2	0.166	0.198	0.237	0.209	0.241	0.112
2	β_1	0.922	0.952	0.937	0.089	0.112	0.096
	β_2	0.607	0.617	0.609	0.091	0.102	0.085
3	β_1	0.857	0.850	0.889	0.102	0.109	0.136
	β_2	0.316	0.333	0.346	0.104	0.123	0.131
	β_1	1.001	1.003	1.005	0.136	0.178	0.135
	β_2	0.891	1.031	1.138	0.144	0.268	0.172
4	β_3	0.521	0.371	0.401	0.071	0.114	0.070
	β_4	1.028	1.108	1.126	0.080	0.158	0.089
	β_5	-0.667	-0.532	-0.547	0.055	0.112	0.067

Table A4.4: Estimation results corresponding to the latency parameters for the leukemia data

p	Parameter	Estimate			Standard Error		
		Logit	Spline	DT	Logit	Spline	DT
2	β_1(Patient Age)	-0.499	-0.403	-0.385	0.276	0.272	0.235
	β_2(Donor Age)	0.664	0.570	0.576	0.282	0.311	0.234
3	β_1(Patient Age)	-0.319	-0.308	-0.299	0.362	0.268	0.295
	β_2(Donor Age)	0.416	0.419	0.405	0.347	0.306	0.302
	β_3(MTX)	0.913	0.899	0.908	0.386	0.323	0.299
4	β_1(Patient Age)	-0.275	-0.283	-0.268	0.331	0.278	0.229
	β_2(Donor Age)	0.338	0.357	0.329	0.321	0.290	0.288
	β_3(MTX)	0.973	0.959	0.976	0.386	0.343	0.290
	β_4(Donor CMV)	0.351	0.325	0.355	0.355	0.349	0.347
7	β_1(Patient Age)	-0.255	-0.291	-0.281	0.319	0.283	0.278
	β_2(Donor Age)	0.428	0.480	0.462	0.321	0.319	0.303
	β_3(Waiting Time)	-0.289	-0.282	-0.271	0.184	0.133	0.149
	β_4(Donor Sex)	-0.071	-0.068	-0.090	0.368	0.350	0.308
	β_5(Patient CMV)	-0.331	-0.324	-0.282	0.425	0.397	0.387
	β_6(Donor CMV)	0.097	0.071	0.106	0.480	0.425	0.389
	β_7(MTX)	1.132	1.106	1.108	0.430	0.380	0.310

Table A4.5: Computation times for the leukemia data

Model	Computation Times (in seconds)			
	p=2	p=3	p=4	p=7
Logit	4.198	4.158	5.224	6.802
Spline	57.703	75.337	82.716	95.871
DT	103.100	138.225	152.410	227.274

Table A4.6: Comparison of DT, NN, and RF models through the estimation of latency parameters for the leukemia data

Model	Estimate		Standard Error		p-value	
	β_1(Patient Age)	β_2(Donor Age)	β_1	β_2	β_1	β_2
DT	-0.385	0.576	0.235	0.234	0.097	0.014
NN	-0.301	0.472	0.234	0.240	0.096	0.036
RF	-0.349	0.520	0.286	0.292	0.145	0.075