APPENDIX 2: R code for BiMM tree and BiMM forest functions

```
#load libraries
library(rpart)
library(blme)
library(randomForest)

#################################################################################
#variable names
#traindata: name of the training dataset
#testdata: name of the test dataset
#formula: formula for fixed variables with binary outcome
       #example:
comagradelow1~Sex+Ethnicity+Age+ALT+AST+Bilirubin+Creat+Phosphate+Lactate+plate
lets+ammonia
              +inr+pressors+rrt
#random: name of the random clustering variable

#################################################################################
#BiMM tree with one iteration
BiMMtree1<-function(traindata,testdata,formula,random){
       #initialize parameters
       minsize=round(length(traindata[,1])/10,0)
       data=traindata
       initialRandomEffects=rep(0,length(data[,1]))
       ErrorTolerance=0.001
       MaxIterations=1000
       tree.control=rpart.control(minbucket=minsize)
       #parse formula
       Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
       TargetName<-formula[[2]]
       Target<-data[,toString(TargetName)]
       #set up variables for loop
       ContinueCondition<-TRUE
       iterations<-0
       #initial values
       AdjustedTarget<-as.numeric(Target)-initialRandomEffects
       oldlik<- -Inf
       # Make a new data frame to include all the new variables
       newdata <- data

       #run 1 iteration of algorithm
              newdata[,"AdjustedTarget"] <- AdjustedTarget
              iterations <- iterations+1
              #build tree
              tree <- rpart(formula(paste(c("AdjustedTarget",
Predictors),collapse = "~")),
                     data = data, method = "class", control = tree.control)

              ## Estimate New Random Effects and Errors using BLMER
              # Get variables that identify the node for each observation
              data[,"nodeInd"] <- 0
              data["nodeInd"] <- tree$where
              # Fit linear model with nodes as predictors (we use the original
target so likelihoods are comparable)
              # Check that the fitted tree has at least two nodes.
       if(min(tree$where)==max(tree$where)){
                     lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),1), collapse="~"),
"+(1|random)",sep=""))),
```

```
	data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")
		} else {
			 lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"as.factor(nodeInd
)"), collapse="~"), "+(1|random)",sep=""))),
				data=data,
family=binomial,control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=200
00000000))),error=function(cond)"skip")
		 }

	#if GLMM did not converge, return NA's for accuracy statistics
	if(class(lmefit)[1]=="character"){
		#return train and test confusion matrices
		return(list(
			c(NA,NA,NA,NA),
			c(NA,NA,NA,NA),
			NA
		))
	}
	else if(!(class(lmefit)[1]=="character")){
		#train dataset predictions
		train.preds.ave<- AdjustedTarget
		train.preds<-predict(tree,traindata,type="class")
		#test dataset predictions
		test.preds<-predict(tree,testdata,type="class")
		#format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
		t1<-table(data$comagradelow,train.preds.ave)
		t4<-table(testdata$comagradelow,test.preds)
		#code if table for train or test data if all predictions are for
same group
		if(ncol(t1)==1 & train.preds.ave[1]==1){
			t1<-c(0,0,t1[1,1],t1[2,1])
		}
		else if(ncol(t1)==1 & train.preds.ave[1]==0){
			t1<-c(t1[1,1],t1[2,1],0,0)
		}
		if(ncol(t4)==1 & test.preds[1]==1){
			t4<-c(0,0,t4[1,1],t4[2,1])
		}
		else if(ncol(t4)==1 & test.preds[1]==0){
			t4<-c(t4[1,1],t4[2,1],0,0)
		}
		#return train and test confusion matrices, # iterations
		return(list(
			c(t1),
			c(t4),
			iterations
		))
	}
}
```

```
#############################################################################
#BiMM forest with one iteration
#note: requires training and test data with no missing values

BiMMforest1<-function(traindata,testdata,formula,random,seed){
	#set up variables for Bimm method
	data=traindata1
	initialRandomEffects=rep(0,length(data[,1]))
	ErrorTolerance=0.006
	MaxIterations=1000
	#parse formula
	Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
	TargetName<-formula[[2]]
	Target<-data[,toString(TargetName)]
	#set up variables for loop
	ContinueCondition<-TRUE
	iterations<-0
	#initial values
	AdjustedTarget<-as.numeric(Target)-initialRandomEffects
	oldlik<- -Inf
	# Make a new data frame to include all the new variables
	newdata <- data

	#compile one iteration of the BiMM forest algorithm
		newdata[,"AdjustedTarget"] <- AdjustedTarget
		iterations <- iterations+1
		#build tree
		set.seed(seed)
		forest <- randomForest(formula(paste(c("factor(AdjustedTarget)",
Predictors),collapse = "~")),
			data = data, method = "class")
		forestprob<-predict(forest,type="prob")[,2]
		## Estimate New Random Effects and Errors using GLMER
		options(warn=-1)
		lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"forestprob"),
collapse="~"), "+(1|random)",sep=""))),

	data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")

	#if GLMM did not converge, produce NAs for accuracy statistics
	if(class(lmefit)[1]=="character"){
		#return train and test confusion matrices
		return(list(
			c(NA,NA,NA,NA),
			c(NA,NA,NA,NA),
			NA
		))
	}
	else if(!(class(lmefit)[1]=="character")){
		test.preds<-predict(forest,testdata1)
		traindata1<-cbind(traindata1,random)
		train.preds<-
ifelse(predict(lmefit,traindata1,type="response")<.5,0,1)
		#format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
		t1<-table(traindata1$comagradelow1,train.preds)
		t4<-table(testdata1$comagradelow1,test.preds)
		if(ncol(t1)==1 & train.preds[1]==1){
```

```
                t1<-c(0,0,t1[1,1],t1[2,1])
            }
            else if(ncol(t1)==1 & train.preds[1]==0){
                t1<-c(t1[1,1],t1[2,1],0,0)
            }
            if(ncol(t4)==1 & test.preds[1]==1){
                t4<-c(0,0,t4[1,1],t4[2,1])
            }
            else if(ncol(t4)==1 & test.preds[1]==0){
                t4<-c(t4[1,1],t4[2,1],0,0)
            }

            #return train and test confusion matrices, # iterations
            return(list(
                c(t1),
                c(t4),
                iterations
            ))
        }
}
```

```
################################################################################
#BiMM tree with H1 updates

BiMMtreeH1<-function(traindata,testdata,formula,random,seed){
	#set up variables for Bimm method
	data=traindata1
	initialRandomEffects=rep(0,length(data[,1]))
	ErrorTolerance=0.006
	MaxIterations=1000
	#parse formula
	Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
	TargetName<-formula[[2]]
	Target<-data[,toString(TargetName)]
	#set up variables for loop
	ContinueCondition<-TRUE
	iterations<-0
	#initial values
	AdjustedTarget<-as.numeric(Target)-initialRandomEffects
	oldlik<- -Inf
	# Make a new data frame to include all the new variables
	newdata <- data

	while(ContinueCondition){
		# Current values of variables
		newdata[,"AdjustedTarget"] <- AdjustedTarget
		iterations <- iterations+1
		#build tree
		tree <- rpart(formula(paste(c("AdjustedTarget",
Predictors),collapse = "~")),
			data = data, method = "class", control = tree.control)

		## Estimate New Random Effects and Errors using BLMER
		# Get variables that identify the node for each observation
		data[,"nodeInd"] <- 0
		data["nodeInd"] <- tree$where
		# Fit linear model with nodes as predictors (we use the original
target so likelihoods are comparable)
		# Check that the fitted tree has at least two nodes.
	if(min(tree$where)==max(tree$where)){
			lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),1), collapse="~"),
"+(1|random)",sep=""))),

	data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")
		} else {
			 lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"as.factor(nodeInd
)"), collapse="~"), "+(1|random)",sep=""))),
				data=data,
family=binomial,control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=200
00000000))),error=function(cond)"skip")
		 }
		# Get the likelihood to check on convergence
		if(!(class(lmefit)[1]=="character")){
			newlik <- logLik(lmefit)
			ContinueCondition <- (newlik-oldlik>ErrorTolerance &
iterations < MaxIterations)
			oldlik <- newlik
			# Extract random effects to make the new adjusted target
```

```
                    logit<-predict(tree,type="prob")[,2]
                    logit2<-
exp(predict(lmefit,re.form=NA))/(1+exp(predict(lmefit,re.form=NA))) #population
level effects
                    AllEffects <- (logit+logit2)/2 #average them
                    #split function h1
                    AdjustedTarget <- ifelse(as.numeric(AdjustedTarget) +
AllEffects>.5,1,0)
             }
             else{ ContinueCondition<-FALSE }
      }

      if(class(lmefit)[1]=="character"){
             #return train and test confusion matrices
             return(list(
                    c(NA,NA,NA,NA),
                    c(NA,NA,NA,NA),
                    NA
             ))
      }
      else if(!(class(lmefit)[1]=="character")){
             #average effects
             train.preds.ave<- AdjustedTarget
             #test dataset predictions-same for all 3 updating methods for the
1 iteration model
             test.preds<-predict(tree,testdata,type="class")
             #format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
             t1<-table(data$ys,train.preds.ave)
             t4<-table(testdata$ys,test.preds)
             if(ncol(t1)==1 & train.preds.ave[1]==1){
                    t1<-c(0,0,t1[1,1],t1[2,1])
             }
             else if(ncol(t1)==1 & train.preds.ave[1]==0){
                    t1<-c(t1[1,1],t1[2,1],0,0)
             }
             if(ncol(t4)==1 & test.preds[1]==1){
                    t4<-c(0,0,t4[1,1],t4[2,1])
             }
             else if(ncol(t4)==1 & test.preds[1]==0){
                    t4<-c(t4[1,1],t4[2,1],0,0)
             }
             #return train and test confusion matrices
             return(list(
                    c(t1),
                    c(t4),
                    iterations
             ))
      }
}
```

```
################################################################################
#BiMM tree with H3 updates

BiMMtreeH3<-function(traindata,testdata,formula,random,seed){
       #set up variables for Bimm method
       data=traindata1
       initialRandomEffects=rep(0,length(data[,1]))
       ErrorTolerance=0.006
       MaxIterations=1000
       #parse formula
       Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
       TargetName<-formula[[2]]
       Target<-data[,toString(TargetName)]
       #set up variables for loop
       ContinueCondition<-TRUE
       iterations<-0
       #initial values
       AdjustedTarget<-as.numeric(Target)-initialRandomEffects
       oldlik<- -Inf
       # Make a new data frame to include all the new variables
       newdata <- data

       while(ContinueCondition){
              # Current values of variables
              newdata[,"AdjustedTarget"] <- AdjustedTarget
              iterations <- iterations+1
              #build tree
              tree <- rpart(formula(paste(c("AdjustedTarget",
Predictors),collapse = "~")),
                     data = data, method = "class", control = tree.control)
              ## Estimate New Random Effects and Errors using BLMER
              # Get variables that identify the node for each observation
              data[,"nodeInd"] <- 0
              data["nodeInd"] <- tree$where
              # Fit linear model with nodes as predictors (we use the original
target so likelihoods are comparable)
              # Check that the fitted tree has at least two nodes.
       if(min(tree$where)==max(tree$where)){
                     lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),1), collapse="~"),
"+(1|random)",sep=""))),

       data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")
              } else {
                      lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"as.factor(nodeInd
)"), collapse="~"), "+(1|random)",sep=""))),
                                 data=data,
family=binomial,control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=200
00000000))),error=function(cond)"skip")
               }
              # Get the likelihood to check on convergence
              if(!(class(lmefit)[1]=="character")){
                     newlik <- logLik(lmefit)
                     ContinueCondition <- (newlik-oldlik>ErrorTolerance &
iterations < MaxIterations)
                     oldlik <- newlik
                     # Extract random effects to make the new adjusted target
                     logit<-predict(tree,type="prob")[,2]
```

```
                    logit2<-
exp(predict(lmefit,re.form=NA))/(1+exp(predict(lmefit,re.form=NA))) #population
level effects
                    AllEffects <- (logit+logit2)/2 #average them
                    #AdjustedTarget <- ifelse(as.numeric(AdjustedTarget) +
AllEffects>.5,1,0)
                    #new split function h3
                    for(k in 1:length(AllEffects)){

      if(as.numeric(AdjustedTarget[k])+AllEffects[k]<.5){AdjustedTarget[k]=0}
                           else
if(as.numeric(AdjustedTarget[k])+AllEffects[k]>1.5){AdjustedTarget[k]=1}
                           else{
                                  #generate random probability coin flip based
on AllEffects (q notation in paper)
                                  AdjustedTarget[k]<-rbinom(1,1,AllEffects[k])
                           }
                    }
             }
             else{ ContinueCondition<-FALSE }
      }

      if(class(lmefit)[1]=="character"){
             #return train and test confusion matrices
             return(list(
                    c(NA,NA,NA,NA),
                    c(NA,NA,NA,NA),
                    NA
             ))
      }
      else if(!(class(lmefit)[1]=="character")){
             #average effects
             train.preds.ave<- AdjustedTarget
             #test dataset predictions-same for all 3 updating methods for the
1 iteration model
             test.preds<-predict(tree,testdata,type="class")
             #format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
             t1<-table(data$ys,train.preds.ave)
             t4<-table(testdata$ys,test.preds)
             if(ncol(t1)==1 & train.preds.ave[1]==1){
                    t1<-c(0,0,t1[1,1],t1[2,1])
             }
             else if(ncol(t1)==1 & train.preds.ave[1]==0){
                    t1<-c(t1[1,1],t1[2,1],0,0)
             }
             if(ncol(t4)==1 & test.preds[1]==1){
                    t4<-c(0,0,t4[1,1],t4[2,1])
             }
             else if(ncol(t4)==1 & test.preds[1]==0){
                    t4<-c(t4[1,1],t4[2,1],0,0)
             }
             #return train and test confusion matrices, # iterations
             return(list(
                    c(t1),
                    c(t4),
                    iterations
             ))
      }
}
```

```
################################################################################
#BiMM forest with H1 updates

BiMMforestH1<-function(traindata,testdata,formula,random,seed){
	#set up variables for Bimm method
	data=traindata1
	initialRandomEffects=rep(0,length(data[,1]))
	ErrorTolerance=0.006
	MaxIterations=1000
	#parse formula
	Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
	TargetName<-formula[[2]]
	Target<-data[,toString(TargetName)]
	#set up variables for loop
	ContinueCondition<-TRUE
	iterations<-0
	#initial values
	AdjustedTarget<-as.numeric(Target)-initialRandomEffects
	oldlik<- -Inf
	# Make a new data frame to include all the new variables
	newdata <- data
	shouldpredict=TRUE

	while(ContinueCondition){
		# Current values of variables
		newdata[,"AdjustedTarget"] <- AdjustedTarget
		iterations <- iterations+1
		#build tree
		set.seed(seed)
		forest <- randomForest(formula(paste(c("factor(AdjustedTarget)",
Predictors),collapse = "~")),
			data = data, method = "class")
		forestprob<-predict(forest,type="prob")[,2]
		## Estimate New Random Effects and Errors using BLMER
		lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"forestprob"),
collapse="~"), "+(1|random)",sep=""))),

	data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")
		# Get the likelihood to check on convergence
		if(!(class(lmefit)[1]=="character")){
			newlik <- logLik(lmefit)
			ContinueCondition <- (abs(newlik-oldlik)>ErrorTolerance &
iterations < MaxIterations)
			oldlik <- newlik
			# Extract random effects to make the new adjusted target
			logit<-forestprob
			logit2<-
exp(predict(lmefit,re.form=NA))/(1+exp(predict(lmefit,re.form=NA))) #population
level effects
			AllEffects <- (logit+logit2)/2 #average them
			#h1 update
			AdjustedTarget <- ifelse(as.numeric(Target) + AllEffects-
1>.5,1,0)

		}
		else{ ContinueCondition<-FALSE }
		#if all of the binary outcomes are the same then get out of loop
		if(min(AdjustedTarget)==max(AdjustedTarget)){
```

```
                        ContinueCondition<-FALSE
                        shouldpredict=FALSE
                }
        }

        if(class(lmefit)[1]=="character" | shouldpredict==FALSE){
                #return train and test confusion matrices
                return(list(
                        c(NA,NA,NA,NA),
                        c(NA,NA,NA,NA),
                        NA,
                        NA
                ))
        }
        else if(!(class(lmefit)[1]=="character")){
                #predictions
                test.preds<-predict(forest,testdata)
                traindata1<-cbind(traindata,random)
                train.preds<-
ifelse(predict(lmefit,traindata1,type="response")<.5,0,1)
                #format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
                t1<-table(traindata$ys,train.preds)
                t4<-table(testdata$ys,test.preds)
                if(ncol(t1)==1 & train.preds[1]==1){
                        t1<-c(0,0,t1[1,1],t1[2,1])
                }
                else if(ncol(t1)==1 & train.preds[1]==0){
                        t1<-c(t1[1,1],t1[2,1],0,0)
                }
                if(ncol(t4)==1 & test.preds[1]==1){
                        t4<-c(0,0,t4[1,1],t4[2,1])
                }
                else if(ncol(t4)==1 & test.preds[1]==0){
                        t4<-c(t4[1,1],t4[2,1],0,0)
                }

                #return train and test confusion matrices, # iterations, and RF
OOBER
                return(list(
                        c(t1),
                        c(t4),
                        iterations,
                        mean(forest$err.rate[,1])
                ))
        }
}
```

```
################################################################################
#BiMM forest with H3 updates

BiMMforestH3<-function(traindata,testdata,formula,random,seed){
	#set up variables for Bimm method
	data=traindata1
	initialRandomEffects=rep(0,length(data[,1]))
	ErrorTolerance=0.006
	MaxIterations=1000
	#parse formula
	Predictors<-paste(attr(terms(formula),"term.labels"),collapse="+")
	TargetName<-formula[[2]]
	Target<-data[,toString(TargetName)]
	#set up variables for loop
	ContinueCondition<-TRUE
	iterations<-0
	#initial values
	AdjustedTarget<-as.numeric(Target)-initialRandomEffects
	oldlik<- -Inf
	# Make a new data frame to include all the new variables
	newdata <- data
	shouldpredict=TRUE

	while(ContinueCondition){
		# Current values of variables
		newdata[,"AdjustedTarget"] <- AdjustedTarget
		iterations <- iterations+1
		#build tree
		set.seed(seed)
		forest <- randomForest(formula(paste(c("factor(AdjustedTarget)",
Predictors),collapse = "~")),
			data = data, method = "class")
		forestprob<-predict(forest,type="prob")[,2]
		## Estimate New Random Effects and Errors using BLMER
		lmefit <-
tryCatch(bglmer(formula(c(paste(paste(c(toString(TargetName),"forestprob"),
collapse="~"), "+(1|random)",sep=""))),

	data=data,family=binomial,control=glmerControl(optCtrl=list(maxfun=20000)
)),error=function(cond)"skip")
		# Get the likelihood to check on convergence
		if(!(class(lmefit)[1]=="character")){
			newlik <- logLik(lmefit)
			ContinueCondition <- (abs(newlik-oldlik)>ErrorTolerance &
iterations < MaxIterations)
			oldlik <- newlik
			# Extract random effects to make the new adjusted target
			logit<-forestprob
			logit2<-
exp(predict(lmefit,re.form=NA))/(1+exp(predict(lmefit,re.form=NA))) #population
level effects
			AllEffects <- (logit+logit2)/2 #average them
			#split function h3
			for(k in 1:length(AllEffects)){
				if(as.numeric(Target[k])+AllEffects[k]-
1<.5){AdjustedTarget[k]=0}
				else if(as.numeric(Target[k])+AllEffects[k]-
1>1.5){AdjustedTarget[k]=1}
				else{
```

```
                                    #generate random probability coin flip based
on AllEffects (q notation in paper)
                                    set.seed(seed)
                                    AdjustedTarget[k]<-rbinom(1,1,AllEffects[k])
                              }
                        }


                }
                else{ ContinueCondition<-FALSE }
                #if all of the binary outcomes are the same then get out of loop
                if(min(AdjustedTarget)==max(AdjustedTarget)){
                        ContinueCondition<-FALSE
                        shouldpredict=FALSE
                }
        }

        if(class(lmefit)[1]=="character" | shouldpredict==FALSE){
                #return train and test confusion matrices
                return(list(
                        c(NA,NA,NA,NA),
                        c(NA,NA,NA,NA),
                        NA,
                        NA
                ))
        }
        else if(!(class(lmefit)[1]=="character")){
                #predictions
                test.preds<-predict(forest,testdata)
                traindata1<-cbind(traindata,random)
                train.preds<-
ifelse(predict(lmefit,traindata1,type="response")<.5,0,1)
                #format table to make sure it always has 4 entries, even if it is
only 2 by 1 (0's in other spots)
                t1<-table(traindata1$ys,train.preds)
                t4<-table(testdata$ys,test.preds)
                if(ncol(t1)==1 & train.preds[1]==1){
                        t1<-c(0,0,t1[1,1],t1[2,1])
                }
                else if(ncol(t1)==1 & train.preds[1]==0){
                        t1<-c(t1[1,1],t1[2,1],0,0)
                }
                if(ncol(t4)==1 & test.preds[1]==1){
                        t4<-c(0,0,t4[1,1],t4[2,1])
                }
                else if(ncol(t4)==1 & test.preds[1]==0){
                        t4<-c(t4[1,1],t4[2,1],0,0)
                }
                #return train and test confusion matrices, # iterations, and RF
OOBER
                return(list(
                        c(t1),
                        c(t4),
                        iterations,
                        mean(forest$err.rate[,1])
                ))
        }
}
```