Norm_Asymp_TrueModNotInSetErrorControlled

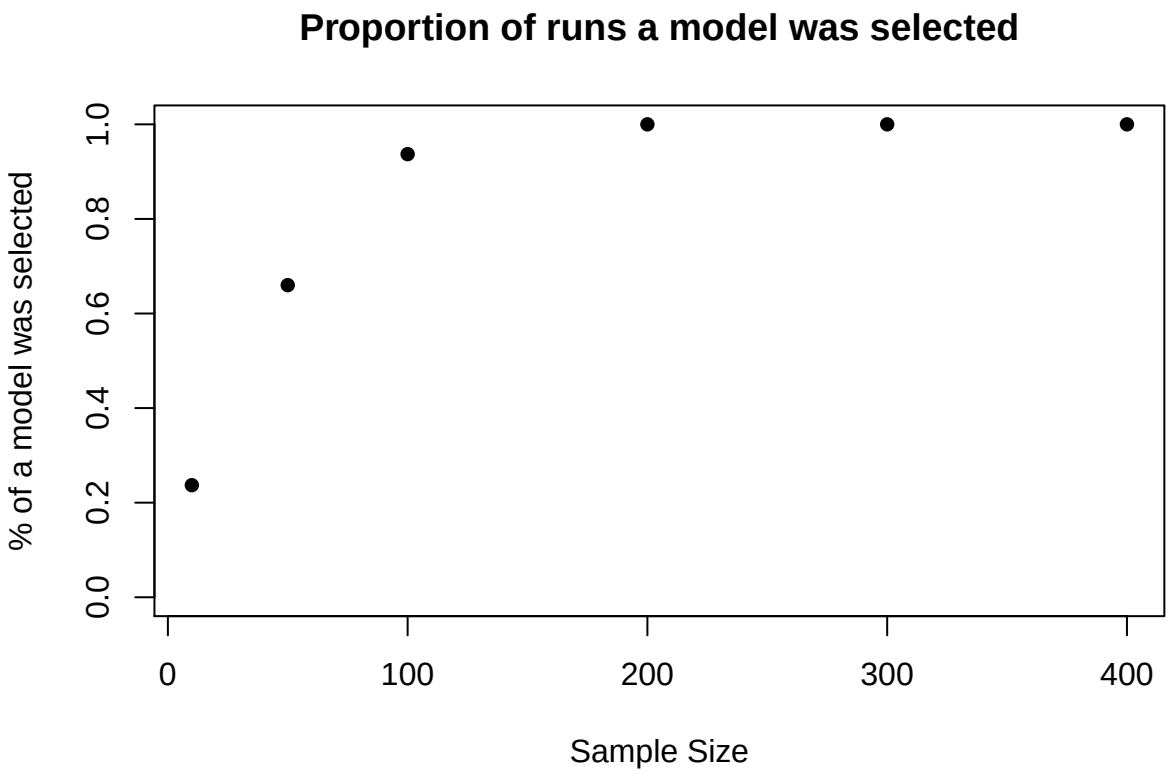
This simulation generates data from a N$(4.3, 1)$ distribution and observes the asymptotic behavior of ECIC using the model set $\mathcal{M} = \{N(3.8, 1), N(4.0, 1), N(4.2, 1), N(4.7, 1)\}$. We see here that the type 1 error rate is still controlled in the short run.

```
######################################################################
#                       Source Function/Data Generation
######################################################################

# source function written in R
source("Simulation_R_Fns.R")
# define the model set
means <- c(3.8,4.0,4.2,4.7)
# set the true mean & true sd
trueMu <- 4.3
trueSig <- 1
trueParams <- c("trueMu"=trueMu,"trueSig"=trueSig)
trueModel <- "N(4.3,1)"
closestModel <- "N(4.2,1)"
M <- list()
MNames <- vector()
# store the cardinality of the model set
for(i in 1:length(means))
{
  tempMean <- means[i]
  M[[i]] <- c("mu"=tempMean,"sig"=trueSig)
  MNames[i] <- paste("N(",tempMean,",",trueSig,")",sep="")
}
MLen <- length(M)
# set different sample sizes
ns <- c(10,50,100,200,300,400)
# store the cardinality of the sample sizes
nsLen <- length(ns)
# set the number of draws for each sample size
noDraws <- 1000
# sample size for estimating the probability of choosing the observed
# best model under the assumption an
# alternative model is true
N1 <- 2000
# sample size for simulating the DGOF distribution under the assumption
# that an alternative model is true
N2 <- 2000
# pre-specified type-1 error rate
alpha <- 0.15
datList <- list()
set.seed(221)
# generate Normal data draws of different sample sizes
```

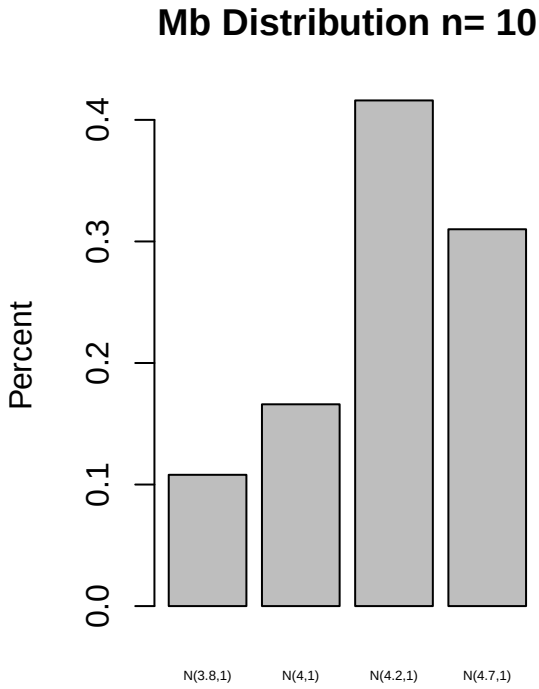
```
for(i in 1:nsLen)
{
  tempN <- ns[i]
  datList[[i]] <- generateData(tempN,noDraws,trueParams,"Normal")
}
# set names for datLists
names(datList) <- paste("True mu=",trueMu,",n=",ns,sep="")

#########################################################################
#                       Run ECIC
#########################################################################

# ECIC step #1
# compute the IC under each model in the model set for each sample size
ICComps <- list()
for(i in 1:nsLen)
{
  ICComps[[i]] <- ICComputations(datList[[i]],M,MLen,noDraws,"NormalNegLL",
                                 MNames)
}
names(ICComps) <- paste("True Model:",trueModel,",Draws of n=",ns,sep="")

# ECIC step #2
# determine the observed best models for each draw of each sample size
MbList <- list()
for(i in 1:nsLen)
{
  MbList[[i]] <- MbComputations(ICComps[[i]],MNames)
}
names(MbList) <- paste("Mbs for ","Draws of n=",ns,sep="")

# ECIC step #3
# compute the observed DGOFs for each draw of each sample size
obsDGOFs <- list()
obsDGOFs <- obsDGOFsComputations(ICComps,nsLen)

# ECIC step #4
# simulate the two data sets necessary for this step
simDat1List <- list() # used to estimate probabilities
simDat2List <- list() # used to estimate DGOF distribution
# initialize the above lists' elements as lists
for(i in 1:nsLen)
{
  simDat1List[[i]] <- list()
  simDat2List[[i]] <- list()
}
set.seed(19)
# simulate data
for(i in 1:nsLen) #  j indexes the assumed true probability
{
  tempN <- ns[i]
  for(j in 1:MLen)
  {
```

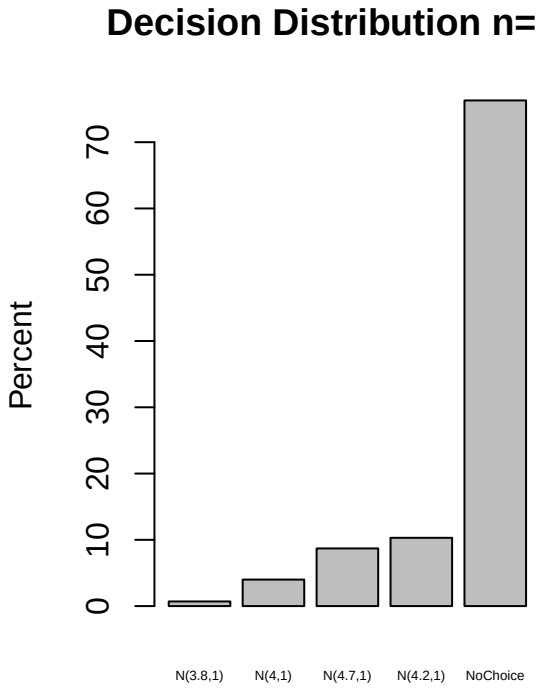
```
    tempM <- M[[j]]
    simDat1List[[i]][[j]] <- generateData(tempN,N1,tempM,"Normal")
    simDat2List[[i]][[j]] <- generateData(tempN,N2,tempM,"Normal")
  }
  names(simDat1List[[i]]) <- paste("True Mod:",MNames,sep="")
  names(simDat2List[[i]]) <- paste("True Mod:",MNames,sep="")
}
names(simDat1List) <- paste("Draws of n=",ns)
names(simDat2List) <- paste("Draws of n=",ns)

# probabilities in the model sets for all draws of each sample size and
# generating probability
ICsSimDat1 <- list()
ICsSimDat2 <- list()
# initialize the above lists' elements as lists
for(i in 1:nsLen)
{
  ICsSimDat1[[i]] <- list()
  ICsSimDat2[[i]] <- list()
}

for(i in 1:nsLen) # i indexes sample sizes
{
  tempN <- ns[i]
  for(j in 1:MLen) # j indexes the assumed true probability
  {
    ICsSimDat1[[i]][[j]] <- ICComputations(simDat1List[[i]][[j]],M,MLen,N1,
                                           "NormalNegLL",MNames)
    ICsSimDat2[[i]][[j]] <- ICComputations(simDat2List[[i]][[j]],M,MLen,N2,
                                           "NormalNegLL",MNames)
  }
  names(ICsSimDat1[[i]]) <- paste("True Mod:",MNames,sep="")
  names(ICsSimDat2[[i]]) <- paste("True Mod:",MNames,sep="")
}
names(ICsSimDat1) <- paste("Draws of n=",ns,sep="")
names(ICsSimDat2) <- paste("Draws of n=",ns,sep="")

# determine the models with the minimum IC for each set of draws
minICList <- list()
for(i in 1:nsLen)
{
  minICList[[i]] <- list()
}

for(i in 1:nsLen) # i indexes sample size
{
  for(j in 1:MLen) # j indexes assumed true parameter
  {
    minICList[[i]][[j]] <- MbComputations(ICsSimDat1[[i]][[j]],MNames)
  }
  names(minICList[[i]]) <- paste("True Mod=",MNames,sep="")
}
names(minICList) <- paste("Draws of n=",ns,sep="")
```

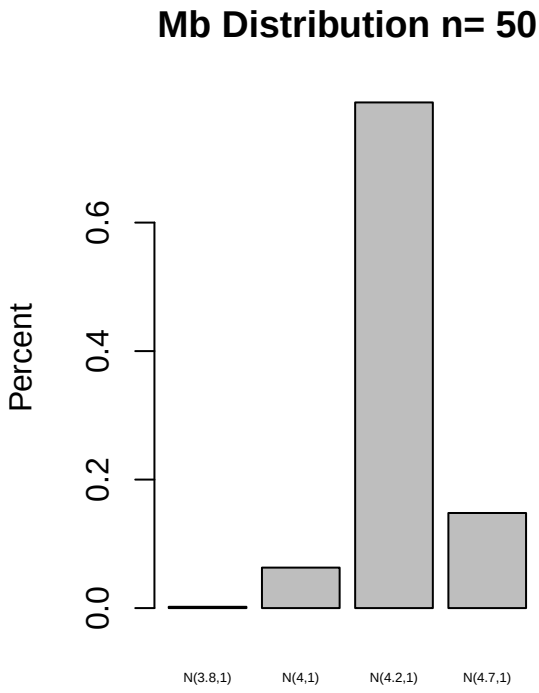
```r
# ECIC step #4b
# create a list of matrices that hold P_i(g(F)=M_b)
piHatList <- list()
for(i in 1:nsLen)
{
  piHatList[[i]] <- piHatMatComputations(minICList[[i]],MLen,N1,MNames)
}
names(piHatList) <- paste("Draws of n=",ns,sep="")

# ECIC step #4c
DGOFList <- list()
for(i in 1:nsLen)
{
  DGOFList[[i]] <- DGOFSimComputations(ICsSimDat2[[i]],mLen,N2,MNames)
  names(DGOFList[[i]]) <- paste("True Model:",MNames,sep="")
}
names(DGOFList) <- paste("Draws of n=",ns,sep="")

# ECIC steps 4d, 5, and 6
resultList <- ECICDecisions(MbList,obsDGOFs,piHatList,DGOFList,alpha,MNames,
                            nsLen,MLen,noDraws)

#########################################################################
#                        Plot Results
#########################################################################

# list to store the decision thresholds
thresholds <- resultList[[1]]
# decision list
aOrRList <- resultList[[2]]

# assess observed best model with ECIC choice
assessList <- list()
for(i in 1:nsLen)
{
  assessList[[i]] <- rbind(MbList[[i]],aOrRList[[i]])
}

DecisionRates <- sapply(X=assessList,FUN=function(x) sum(as.numeric(x[2,]))/
                          noDraws)
plot(x=ns,y=DecisionRates,main="Proportion of runs a model was selected",
     xlab="Sample Size",ylab="% of a model was selected",pch=16,ylim = c(0,1))
```

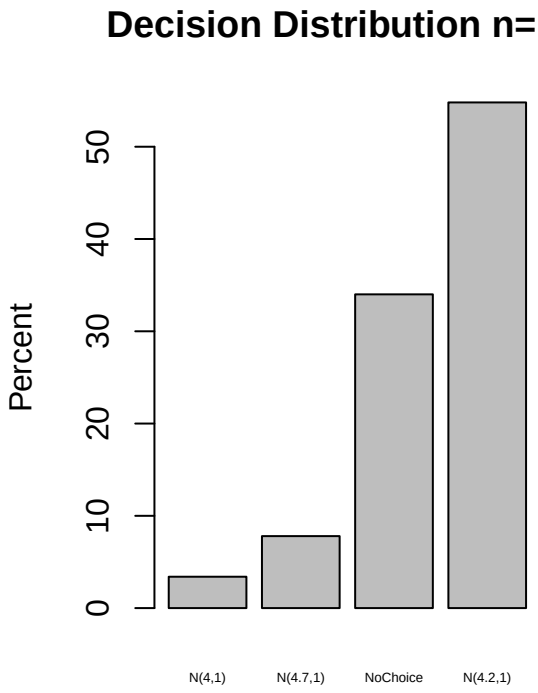


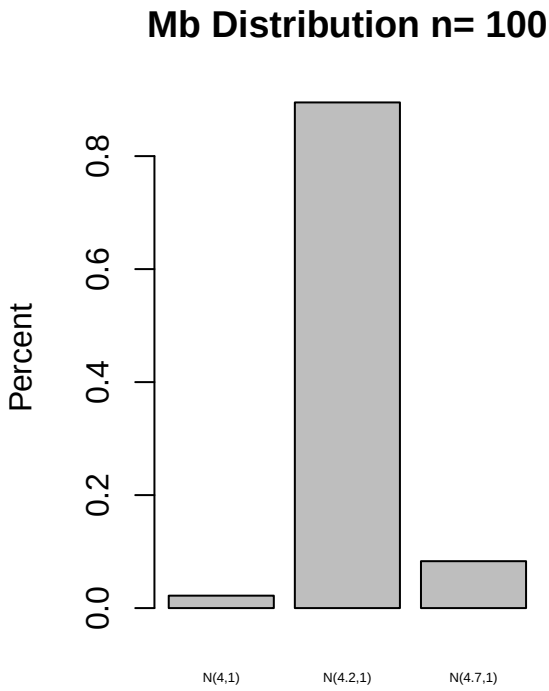

```
# take a look at type 1 error rate
# subset assess list by only when a decision was made i.e. second row = 1
decAssessList <- lapply(X=assessList,FUN=function(x) x[,x[2,]==1])
# plot the model decision distribution
# make barplots for the percentage of the time each model was chosen as best
# for each sample size
par(mfrow = c(1, 2))
for(i in 1:nsLen)
{
  # get the frequencies of models selected as best
  tempTable <- table(MbList[[i]])
  barplot(tempTable/noDraws,main=paste("Mb Distribution n=",ns[i]),
          cex.names=0.45,ylab="Percent")
  chosenModelRate <- table(decAssessList[[i]][1,])/noDraws*100
  unChosenModelRate <- (noDraws-ncol(decAssessList[[i]]))/noDraws*100
  decDist <- c(chosenModelRate,"NoChoice"=unChosenModelRate)
  decDist <- sort(decDist)
  barplot(decDist,main=paste("Decision Distribution n=",ns[i]),cex.names=0.45,
          ylab="Percent")
}
```

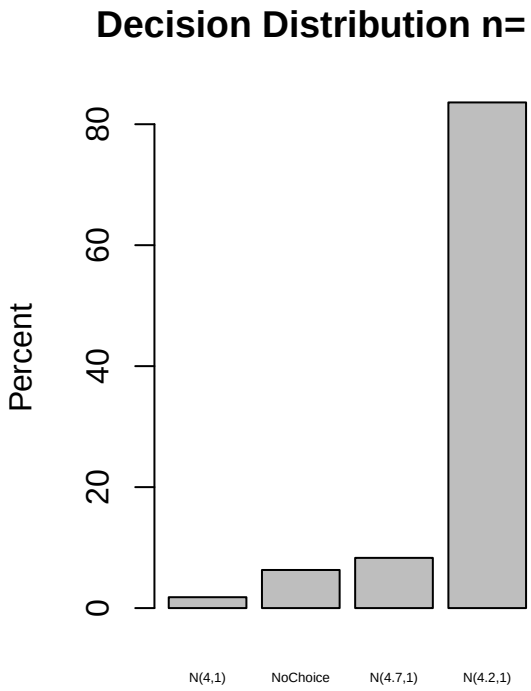

Mb Distribution n= 10
Percent
0.0
0.1
0.2
0.3
0.4
N(3.8,1)
N(4,1)
N(4.2,1)
N(4.7,1)

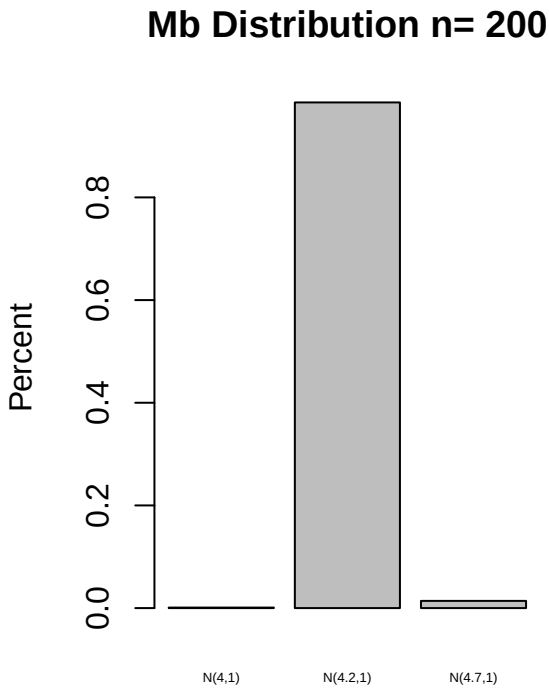

Decision Distribution n= 10
Percent
0
10
20
30
40
50
60
70
N(3.8,1)
N(4,1)
N(4.7,1)
N(4.2,1)
NoChoice

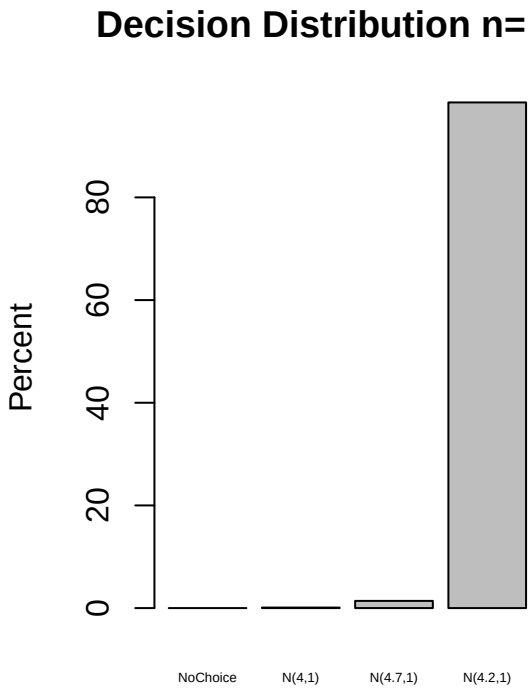

Mb Distribution n= 50
Percent
0.0
0.2
0.4
0.6
N(3.8,1)
N(4,1)
N(4.2,1)
N(4.7,1)

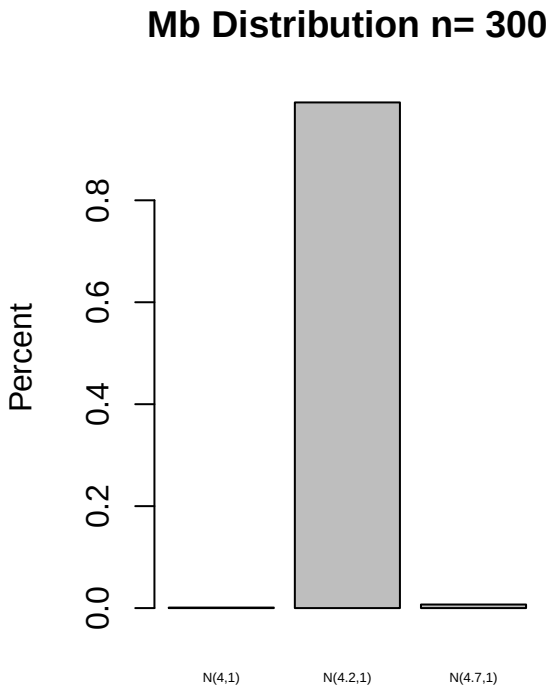

Decision Distribution n= 50
Percent
0
10
20
30
40
50
N(4,1)
N(4.7,1)
NoChoice
N(4.2,1)

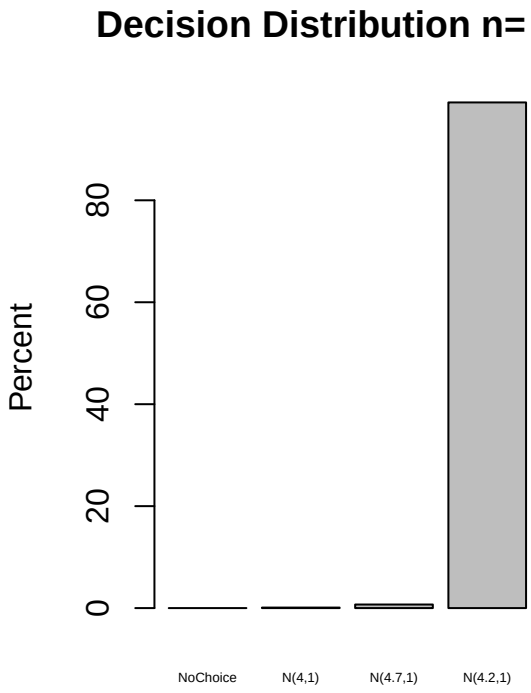

Mb Distribution n= 100
Percent
0.0
0.2
0.4
0.6
0.8
N(4,1)
N(4.2,1)
N(4.7,1)

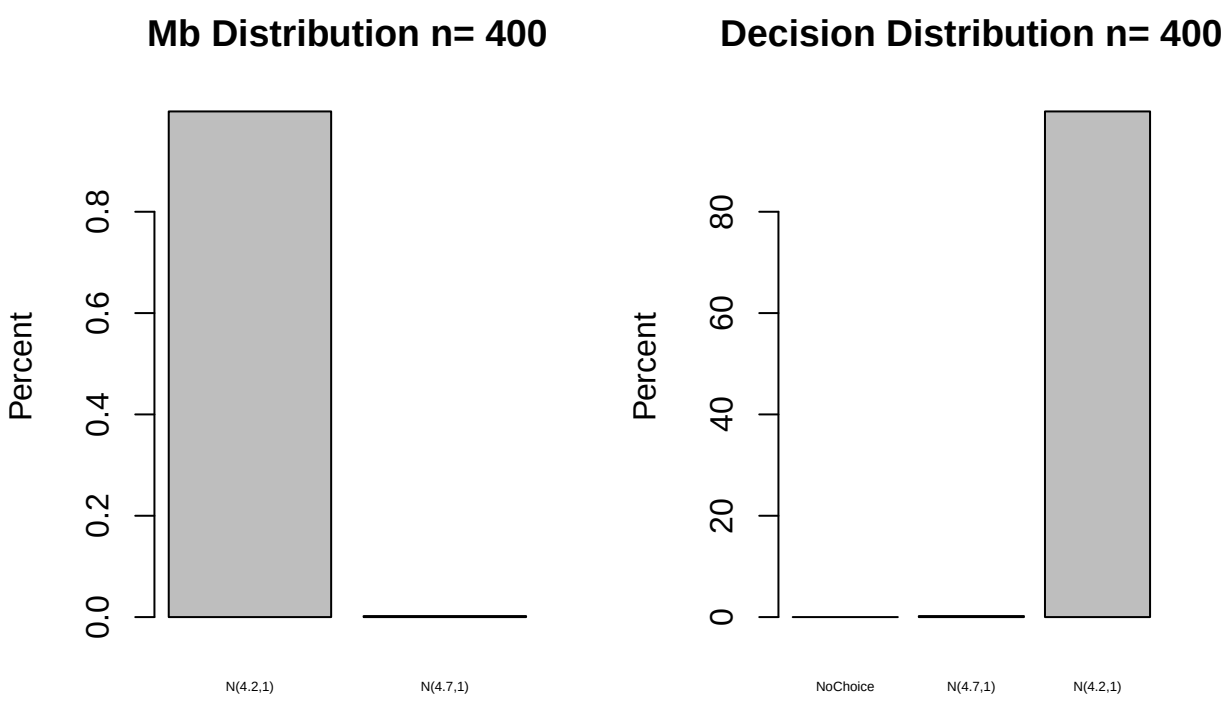

Decision Distribution n= 100
Percent
0
20
40
60
80
N(4,1)
NoChoice
N(4.7,1)
N(4.2,1)


Mb Distribution n= 200
Percent
0.0
0.2
0.4
0.6
0.8
N(4,1)
N(4.2,1)
N(4.7,1)

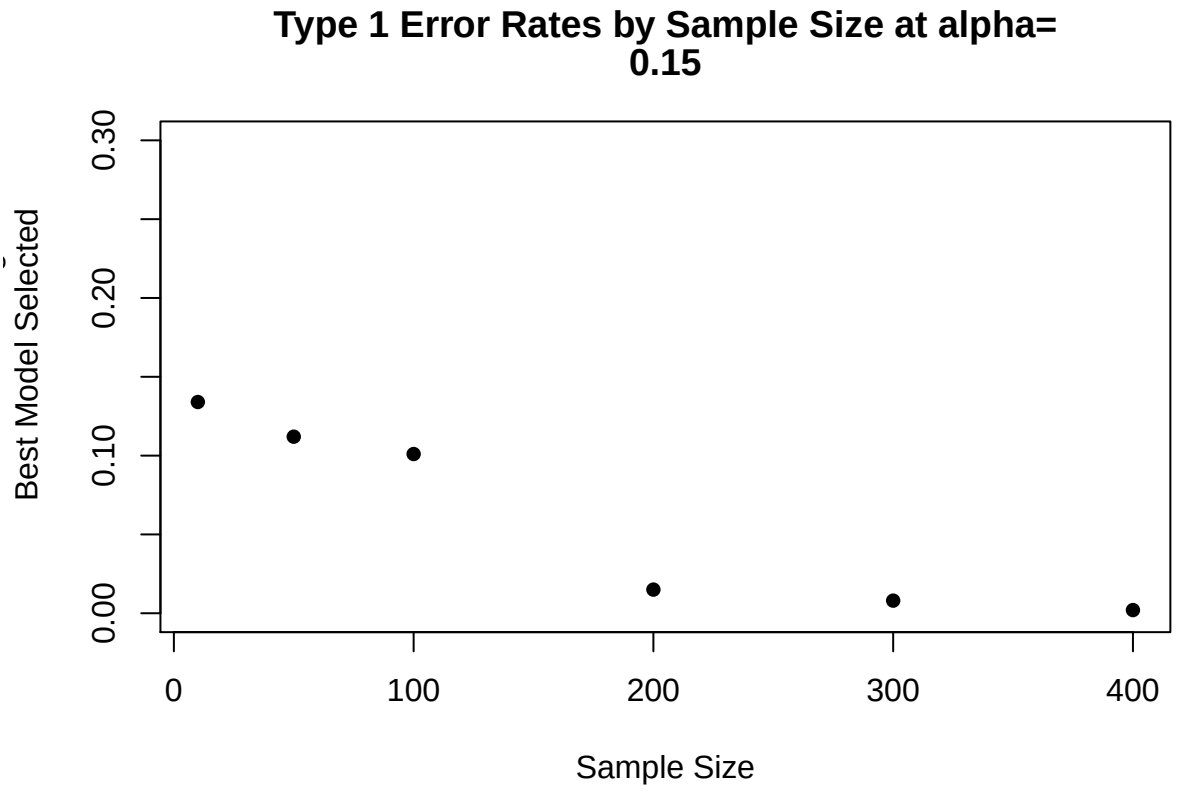

Decision Distribution n= 200
Percent
0
20
40
60
80
NoChoice
N(4,1)
N(4.7,1)
N(4.2,1)


Mb Distribution n= 300
Percent
0.0
0.2
0.4
0.6
0.8
N(4,1)
N(4.2,1)
N(4.7,1)

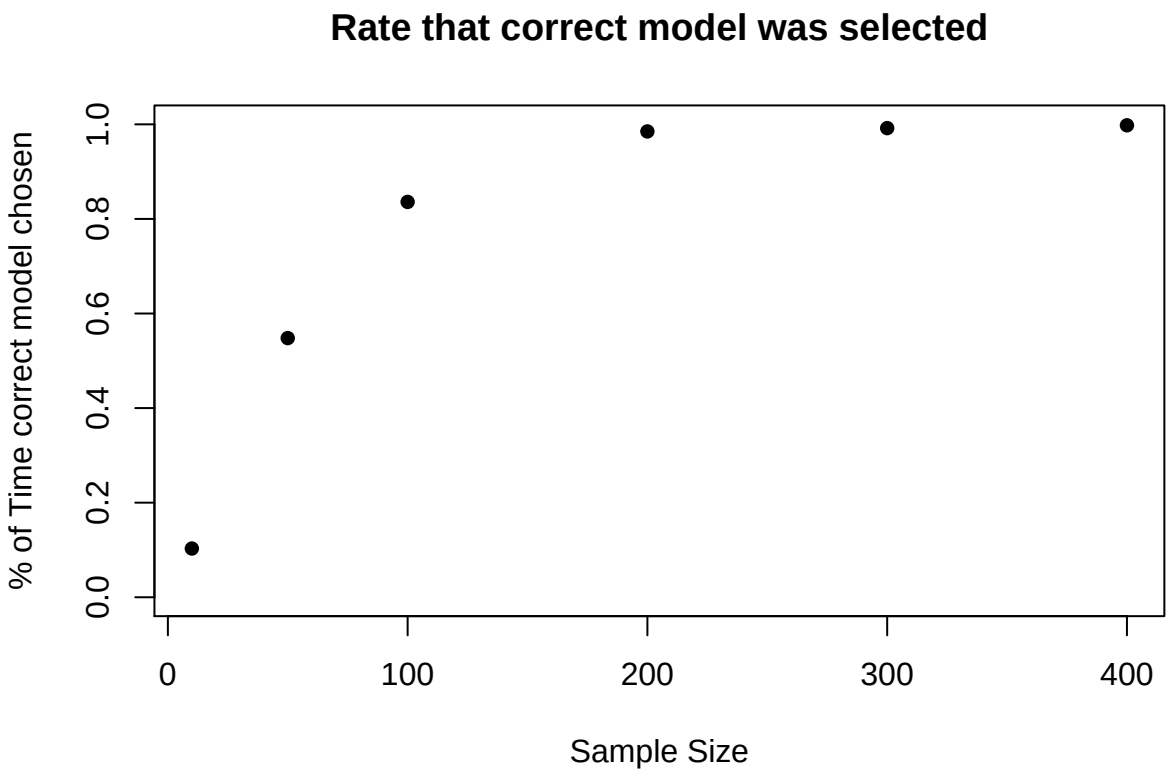

Decision Distribution n= 300
Percent
0
20
40
60
80
NoChoice
N(4,1)
N(4.7,1)
N(4.2,1)




```
par(mfrow = c(1, 1))
# compute type 1 error rate
t1ErrorRates <- sapply(X=decAssessList,FUN=function(x) sum(x[1,]!=closestModel)/
                         noDraws)
plot(x=ns,y=t1ErrorRates,main=c("Type 1 Error Rates by Sample Size at alpha=",
                                alpha), xlab="Sample Size",ylab="% of Time Wrong
     Best Model Selected",pch=16,ylim = c(0,2*alpha))
```




```
CorrectRates <- sapply(X=decAssessList,FUN=function(x) sum(x[1,]==closestModel)/
                         noDraws)
plot(x=ns,y=CorrectRates,main="Rate that correct model was selected",
     xlab="Sample Size",ylab="% of Time correct model chosen",pch=16,
     ylim = c(0,1))
```


Rate that correct model was selected
% of Time correct model chosen
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
Sample Size