Norm_Asymp_WithMLE_LargePenalty

This file runs the same program as in Norm_Asymp_WithMLE except that the only sample size considered is 500, the model set is reduced to $\mathcal{M} = \{N(\mu, 1), N(\mu, \sigma)\}$, and a a multiplicative factor of 10 applied to the penalty term on the BIC calculations. The objective was to observe if the rate of model selection and correct model selection increase to 1 asymptotically, and it appears that which appears to be true. The adjustment to the penalty term for the BIC was used as a proxy for increasing the sample size to larger values (say 10K) due to RAM constraints

```
######################################################################
#                       Source Functions/Data Generation
######################################################################

library(Rcpp)
```

```
## Warning: package 'Rcpp' was built under R version 4.0.5
```

```
library(RcppArmadillo)
```

```
## Warning: package 'RcppArmadillo' was built under R version 4.0.5
```

```
# load C++ script that is used for simulation steps in ECIC
sourceCpp("FnsInCPlusPlus\\Simulation_Rcpp_Fns_Norm.cpp")
# load function in written in R
source("FnsInR\\Simulation_R_Fns.R")
# set true normal model parameters
trueMu <- 0.5
trueSig <- 1
trueParams <- c("trueMu"=trueMu,"trueSig"=trueSig)
trueModel <- "N(0.5,1)"
MNames <- c("N(mu,1)","N(mu,sigma)") #c("N(0,1)","N(mu,1)","N(mu,sigma)")
# model that should be selected
closestMod <- "N(mu,1)"
# cardinality of the model set
MLen <- length(MNames)
M <- list()
for(i in 1:MLen)
{
  M[[i]] <- MNames[i]
}
# set different sample sizes
ns <- 500   # c(3,10,50,100,200)
# cardinality of the sample sizes
nsLen <- length(ns)
# set the number of draws for each sample size
noDraws <- 100
```

```r
# sample size for estimating the probability of choosing the observed best
# model under the assumption an
# alternative model is true
N1 <- 200
# sample size for simulating the DGOF distribution under the assumption that
# an alternative model is true
N2 <- 700
# pre-specified type-1 error rate
alpha <- 0.15
datList <- list()
set.seed(225)
# generate data from the true normal distribution
for(i in 1:nsLen)
{
  tempN <- ns[i]
  datList[[i]] <- generateData(tempN,noDraws,trueParams,"Normal")
}

# compute MLE estimates from the generated data
MLEList <- list()
for(i in 1:nsLen)
{
  MLEList[[i]] <- list()
}

for(i in 1:nsLen)
{
  tempN <- ns[i]
  tempDatList <- datList[[i]]
  tempMeanMLEs <- apply(X=tempDatList,MARGIN=2,FUN=function(x) mean(x))
  tempSigMLEs <- apply(X=tempDatList,MARGIN=2,FUN=function(x) sd(x))
  # convert from unbiased estimate to MLE
  tempSigMLEs <- tempSigMLEs*sqrt((tempN-1))/sqrt(tempN)
  # MLE's for N(mu,1)
  MLEList[[i]][[1]] <- rbind(tempMeanMLEs,1)
  rownames(MLEList[[i]][[1]]) <- c("MLEMean","fixedSig")
  # MLE's for N(mu,sigma)
  MLEList[[i]][[2]] <- rbind(tempMeanMLEs,tempSigMLEs)
  rownames(MLEList[[i]][[2]]) <- c("MLEMean","MLESig")
  colnames(MLEList[[i]][[1]]) <- colnames(MLEList[[i]][[2]]) <-
    colnames(MLEList[[i]][[2]]) <- paste("Draw",1:noDraws,sep="")
  names(MLEList[[i]]) <- paste("MLEs for Model",MNames)
}
names(MLEList) <- paste("Draws of n=",ns,sep="")

######################################################################
#             Begin 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)
```

```r
{
  ICComps[[i]] <- ICComputations(datMat=datList[[i]],M=M,MLen=MLen,
                                 noDraws=noDraws,
                                 ICType="BICNormLP",MNames=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
set.seed(19)
ptm <- proc.time() #Start timing
simDat1List <- normDatSimRcpp(ns,MLEList,N1)
```

```
## Iteration 1 Complete
```

```r
simDat2List <- normDatSimRcpp(ns,MLEList,N2)
```

```
## Iteration 1 Complete
```

```r
proc.time() - ptm
```

```
##    user  system elapsed
##    2.13    0.11    2.08
```

```r
names(simDat1List) <- paste("Draws of n=",ns,sep="")
names(simDat2List) <- paste("Draws of n=",ns,sep="")
# provide names for both simulated sets
for(i in 1:nsLen)
{
  for(j in 1:MLen) # indexes the models in the model set
  {
    names(simDat1List[[i]][[j]]) <- paste("Obs",1:noDraws,";",N1,
                                          "Simulated Draws")
    names(simDat2List[[i]][[j]]) <- paste("Obs",1:noDraws,";",N2,
                                          "Simulated Draws")
  }
  names(simDat1List[[i]]) <- paste("Generated from",MNames)
  names(simDat2List[[i]]) <- paste("Generated from",MNames)
```

```r
}

# simulate distributions to estimate probabilities
ptm=proc.time()
ICsSimDat1 <- ICCompsRcpp(simDat1List,MNames)
```

```
## Its. for Sample Size Index 1 completed
```

```r
proc.time() - ptm
```

```
##    user  system elapsed
##    1.07    0.17    1.29
```

```r
# label the elements in ICsSimDat1
for(i in 1:nsLen)
{
  for(j in 1:MLen)
  {
    for(k in 1:noDraws)
    {
      colnames(ICsSimDat1[[i]][[j]][[k]]) <- paste("BIC Under", MNames)
    }
    names(ICsSimDat1[[i]][[j]]) <- paste("Normal Fit for Obs",1:noDraws)
  }
  names(ICsSimDat1[[i]]) <- paste("Generated from",MNames)
}
names(ICsSimDat1) <- paste("Draws of n=",ns,sep="")

# determine the model with the minimum IC for each set of draws
minICList <- list()
for(i in 1:nsLen)
{
  minICList[[i]] <- list()
  for(j in 1:MLen)
    minICList[[i]][[j]] <- list()
}
for(i in 1:nsLen) # i indexes sample size
{
  for(j in 1:MLen) # j indexes assumed true parameter
  {
    for(k in 1:noDraws)
    {
      minICList[[i]][[j]][[k]] <- apply(ICsSimDat1[[i]][[j]][[k]],
                                        MARGIN=1,FUN=function(x)
                                          MNames[which.min(x)])
    }
    names(minICList[[i]][[j]]) <- paste("Normal Fit for Obs",1:noDraws)
  }
  names(minICList[[i]]) <- paste("Generated from",MNames)
  print(i)
}
```

```
## [1] 1
```

```
names(minICList) <- paste("Draws of n=",ns,sep="")

# 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]] <- list()
  for(k in 1:noDraws)
  {
    piHatList[[i]][[k]] <- list()
  }
  names(piHatList[[i]]) <- paste("Normal Fit for Obs",1:noDraws)
}
names(piHatList) <- names(piHatList) <- paste("n=",ns,sep="")

# compute the probabilities
for(i in 1:nsLen) # indexes the sample size
{
  piHatList[[i]] <- piHatMatComputationsMLEs(minICList[[i]],MLen,N1,
                                              MNames,noDraws)
  print(i)
}
```

```
## [1] 1
```

```
names(piHatList) <- paste("n=",ns,sep="")

rm(ICsSimDat1)
rm(minICList)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells   612233  32.7    1350812  72.2  1350812  72.2
## Vcells 91177273 695.7  130856568 998.4 91813776 700.5
```

```
# ECIC step #4c
DGOFList <- simDGOFsRcpp(simDat2List,MNames)
```

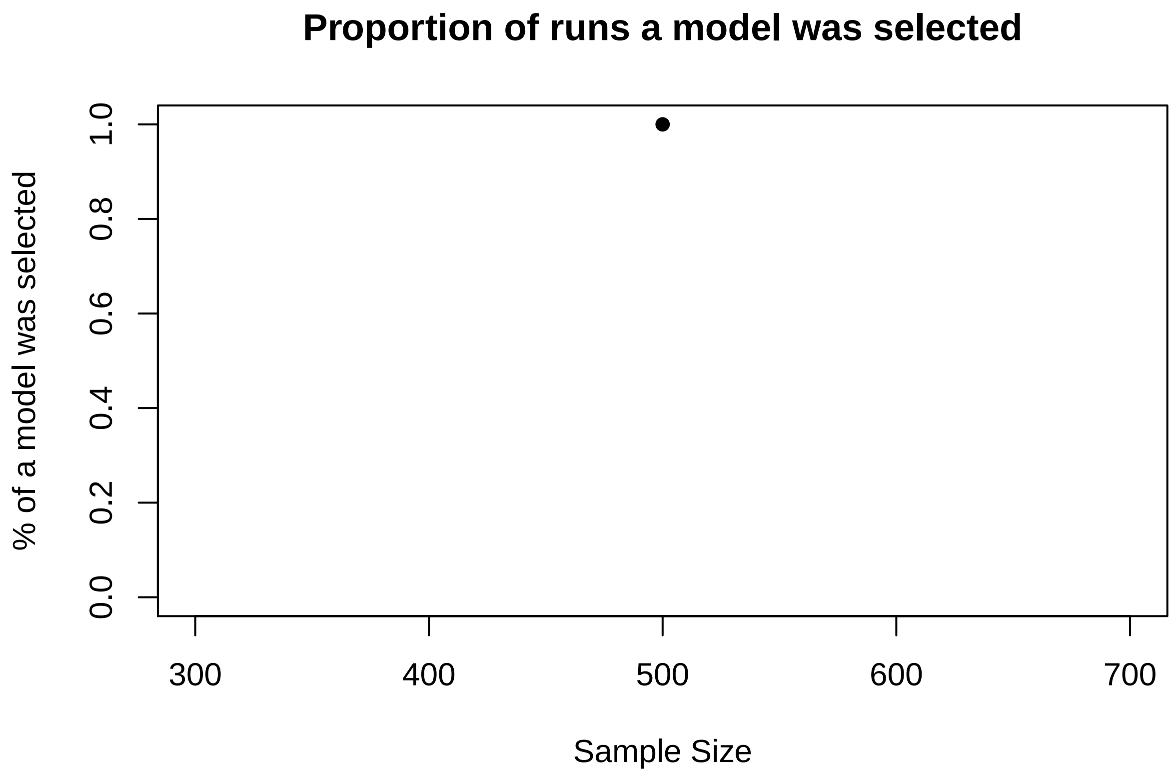
```
## Its. for Sample Size Index 1 completed
```

```
# label the elements in DGOFList
for(i in 1:nsLen)
{
  for(j in 1:MLen)
  {
    for(k in 1:noDraws)
    {
      colnames(DGOFList[[i]][[j]][[k]]) <- paste("DGOF Under", MNames,
                                                  " Observed Best")
    }
```

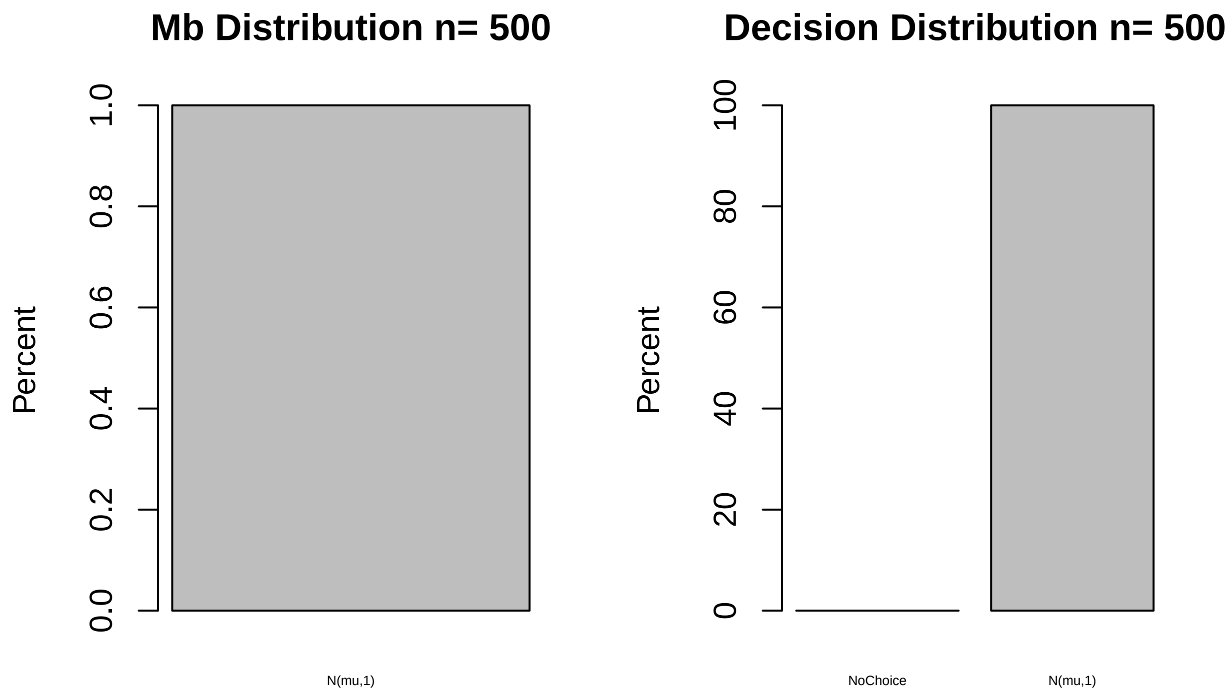
```
    names(DGOFList[[i]][[j]]) <- paste("Normal Fit for Obs",1:noDraws)
  }
  names(DGOFList[[i]]) <- paste("Generated from",MNames)
}
names(DGOFList) <- paste("Draws of n=",ns,sep="")

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

```
## [1] 1
```

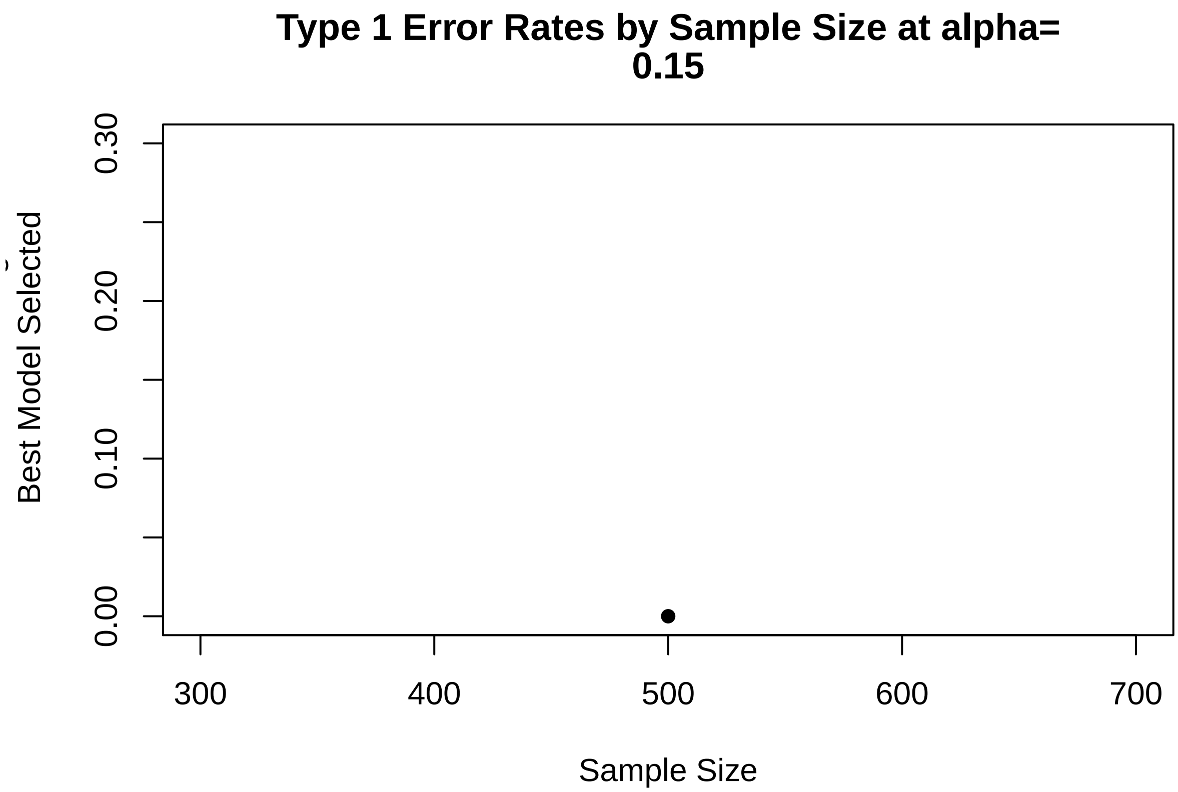
```
######################################################################
#                      Plot Results
######################################################################

# 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))
```



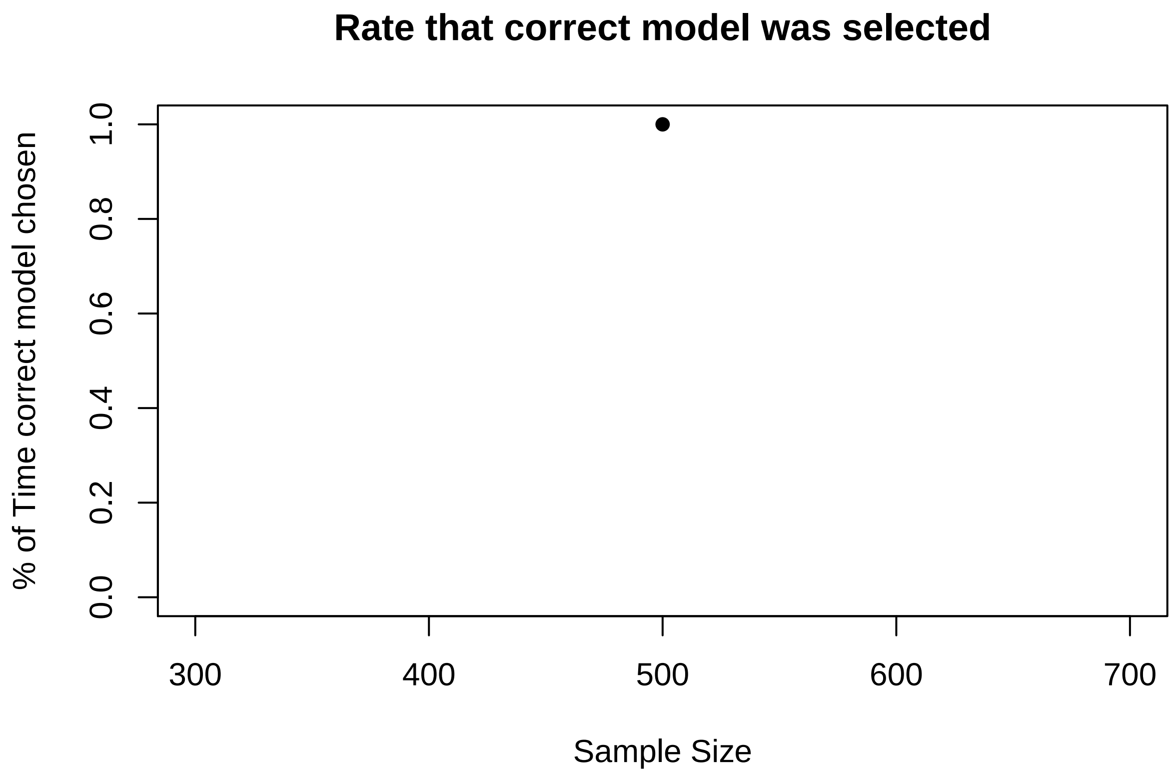

```
# take a look at type 1 error rate
# subset assess list by only when a decision was made i.e. second row = 1
decAssessList <- lapply(X=assessList,FUN=function(x) x[,x[2,]==1])
par(mfrow = c(1, 2))
for(i in 1:nsLen)
{
  # get the frequencies of models selected as best
  tempTable <- table(MbList[[i]])
  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")
}
```




```
par(mfrow = c(1, 1))
# compute type 1 error rate
t1ErrorRates <- sapply(X=decAssessList,FUN=function(x) sum(x[1,]!=closestMod)/
                        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,]==closestMod)/
                         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
300
400
500
600
700
Sample Size