Spline_Asymp_TrueModInSet_WithMLE_NoBiasCorrect

This simulation generates data from a cubic spline observed on $[-10, 10]$ with knots at the quintiles of this interval, a coefficient vector of $\boldsymbol{\beta} = (1, -1, 1, -1, 1, -1, 1)$, and an error standard deviation of 0.06. We observe the asymptotic behavior of ECIC using a model set with knots at varying perceniles on $[-10, 10]$. Specifically we have $\mathcal{M}$={Knots at Quintiles,Knots at Octiles,Knots at Deciles,Knots at Dodeciles}

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

library(Rcpp)
```

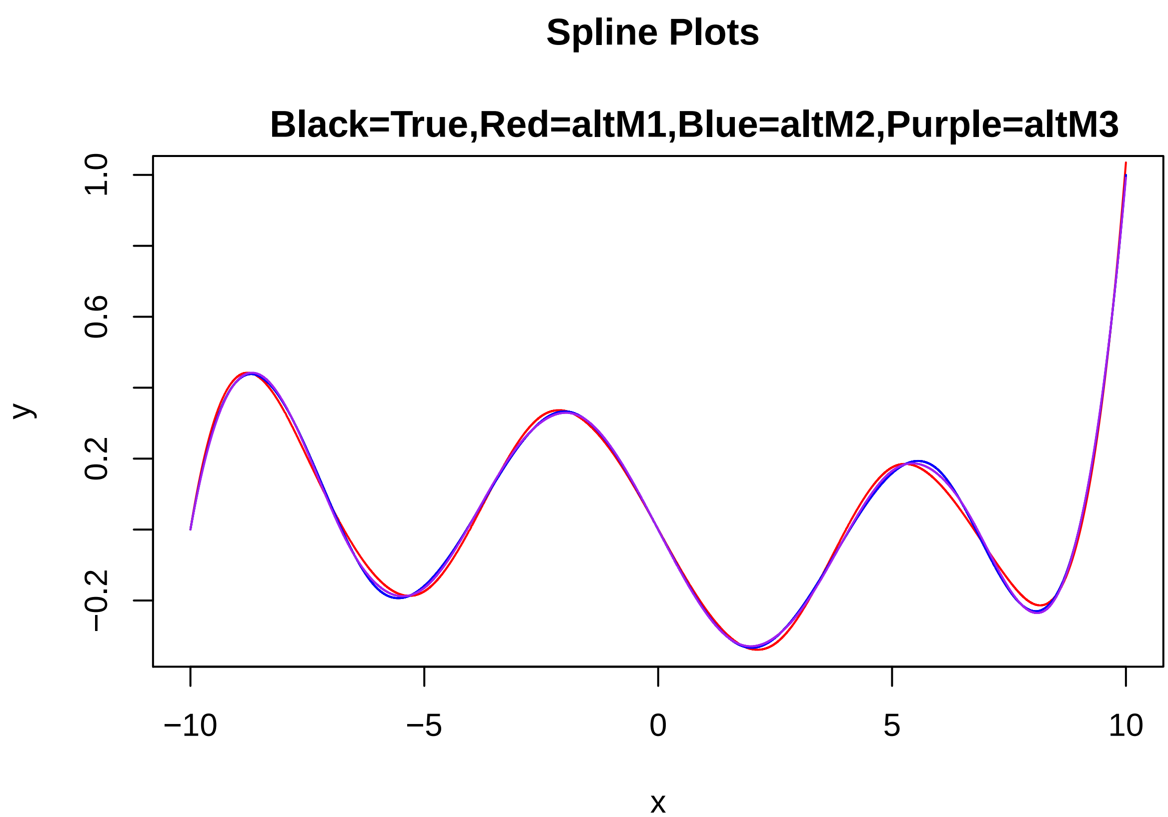
```
## Warning: package 'Rcpp' was built under R version 4.0.5
```

```
library(RcppArmadillo)
```

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

```
library(splines)
# load C++ script that is used for simulation steps in ECIC
sourceCpp("FnsInCPlusPlus\\Simulation_Rcpp_Fns_Splines.cpp")
# load function in written in R
source("FnsInR\\Simulation_R_Fns.R")
# specify the interval over which the splines will be simulated
lowLim <- -10
upLim <- 10
# vector of saturated time points to plot the true spline and
# alternative models
manyX <- seq(from=lowLim,to=upLim,length.out=50000)
# true knots will be placed at the quintiles
trueKnots <- quantile(-10:10,probs=c(1/5,2/5,3/5,4/5))
trueMod <- "quintileKnots"
# create a B-spline basis to plot the true spline
manyBase <- bs(manyX,knots=trueKnots,degree=3)
# set the true regression coefficients
trueCoef <- c(1,-1,1,-1,1,-1,1)
# evaluate the true spline's response values
manyY <- manyBase%*%trueCoef
# plot the true spline
plot(manyX,manyY,type="l",main=paste("Spline Plots \n
       Black=True,Red=altM1,Blue=altM2,Purple=altM3"),
     xlab="x",ylab="y")
# set up bases for alternative models
altM1Knots <- quantile(-10:10,probs=1:7/8) # knot at octiles
altM2Knots <- quantile(-10:10,probs=1:9/10) # knots at deciles
```

```r
altM3Knots <- quantile(-10:10,probs=1:11/12) # knots at dodeciles
# create model set of knot locations for the true and alternative models
# NOTE: we fix the knots locations & degree
# but not the regression coefficients for the alternative models
M <- list("quintileKnots"=trueKnots,"octileKnots"=altM1Knots,
          "decileKnots"=altM2Knots,"dodecileKnots"=altM3Knots)
MLen <- length(M)
MNames <- names(M)
# fit lm using alternative bases to manyY and manyX to get an idea of how these
# models compare
lmAltM1 <- lm(manyY~bs(manyX,knots=altM1Knots,degree=3)-1)
lmAltM2 <- lm(manyY~bs(manyX,knots=altM2Knots,degree=3)-1)
lmAltM3 <- lm(manyY~bs(manyX,knots=altM3Knots,degree=3)-1)
lines(manyX,lmAltM1$fitted.values,col="red")
lines(manyX,lmAltM2$fitted.values,col="blue")
lines(manyX,lmAltM3$fitted.values,col="purple")
```




```r
rm(lmAltM1)
rm(lmAltM2)
rm(lmAltM3)
rm(manyY)
rm(manyBase)
gc()
```

```
##             used (Mb) gc trigger (Mb) max used (Mb)
```

```
## Ncells  624276 33.4    1354535 72.4  1354535 72.4
## Vcells 2424459 18.5   10146329 77.5 10108147 77.2
```

```
# set different sample sizes
ns <- c(15,18,25,30,60)
# store the cardinality of the sample sizes
nsLen <- length(ns)
# set the number of draws for each sample size
noDraws <- 500
# sample size for estimating the probability of choosing the observed
# best model under the assumption an
# alternative model is true
N1 <- 500
# sample size for simulating the DGOF distribution under the
# assumption that an alternative model is true
N2 <- 900
# pre-specified type-1 error rate
alpha <- 0.15
# noise added to generate data
trueSig <- 0.06
# true error paramaters
trueErrorParams <- c(0,trueSig)
# create a list to store the observed points along the x axis
# for each sample size
xPoints <- list()
basisList <- list()
for(i in 1:nsLen)
{
  basisList[[i]] <- list()
}

for(i in 1:nsLen)
{
  xPoints[[i]] <- seq(from=lowLim,to=upLim,length.out=ns[i])
}

for(i in 1:nsLen)
{
  for(j in 1:MLen)
  {
    basisList[[i]][[j]] <- bs(xPoints[[i]],knots=M[[j]],degree=3)
  }
}

datList <- list()
set.seed(222)
for(i in 1:nsLen)
{
  tempXPoints <- xPoints[[i]]
  tempN <- ns[i]
  tempMat <- matrix(NA,nrow=tempN,ncol=noDraws)
  tempBasisVals <- bs(tempXPoints,knots=trueKnots,degree=3)
  tempYVals <- tempBasisVals%*%trueCoef
  # create a matrix of true YVals
```

```r
  tempYVals <- matrix(rep(tempYVals,noDraws),nrow=tempN,ncol=noDraws)
  tempMat <- tempYVals + generateData(tempN,noDraws,trueErrorParams,"Normal")
  colnames(tempMat) <- paste("Draw",1:noDraws,sep="")
  datList[[i]] <- tempMat
}
names(datList) <- 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
lmFitList <- list()
for(i in 1:nsLen)
{
  lmFitList[[i]] <- list()
  for(j in 1:MLen)
    lmFitList[[i]][[j]] <- list()
}
# fit a cubic spline to the data
for(i in 1:nsLen) # i indexes sample size
{
  tempDat <- datList[[i]]
  tempX <- xPoints[[i]]
  for(j in 1:MLen) #j indexes the model
  {
    tempBasis <- bs(tempX,knots=M[[j]],degree=3)
    lmFitList[[i]][[j]] <- apply(X=tempDat,MARGIN=2,
                                 FUN=function(x) lm(x~tempBasis-1))
  }
  names(lmFitList[[i]]) <- MNames
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

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

# now compute the BIC under each model
ICComps <- list()
for(i in 1:nsLen)
{
  ICComps[[i]] <- list()
}

for(i in 1:nsLen) # i indexes the sample size
{
  tempMat <- matrix(data=NA,nrow=MLen,ncol=noDraws)
```

```
  rownames(tempMat) <- paste("BICs",MNames)
  for(j in 1:MLen) #j indexes the model
  {
    tempBICs <- sapply(lmFitList[[i]][[j]],FUN=function(x) BIC(x))
    tempMat[j,] <- tempBICs
  }
  ICComps[[i]] <- tempMat
}
names(ICComps) <- paste("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("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)

# clear up some memory
rm(ICComps)
gc()
```

```
##            used (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  1710027 91.4    3039667 162.4  3039667 162.4
## Vcells 12370326 94.4   21618257 165.0 17948546 137.0
```

```
# ECIC step #4
# simDat1List holds the datasets to estimate \hat{\pi_i}=P_i(g(F)=M_b)
# simDat2List holds the datasets to estimate the DGOF distributions \Delta f_i
simDat1List <- list()
simDat2List <- list()
# initialize the above lists' elements as lists
for(i in 1:nsLen)
{
  simDat1List[[i]] <- list()
  simDat2List[[i]] <- list()
  for(j in 1:MLen)
  {
    simDat1List[[i]][[j]] <- list()
    simDat2List[[i]][[j]] <- list()
  }
}

set.seed(19)
ptm <- proc.time()
# create matrices of random errors using \hat{\sigma} for each model
for(i in 1:nsLen) # index the sample sizes
```

```
{
  tempN <- ns[i]
  for(j in 1:MLen) # indexes the models in the model set
  {
    simDat1List[[i]][[j]] <- lapply(X=lmFitList[[i]][[j]]
                                    ,FUN=function(x) lmGenDatWMLE(x,tempN,N1))
    simDat2List[[i]][[j]] <- lapply(X=lmFitList[[i]][[j]]
                                    ,FUN=function(x) lmGenDatWMLE(x,tempN,N2))
    names(simDat1List[[i]][[j]]) <- paste("lm Fit for Obs",1:noDraws)
    names(simDat2List[[i]][[j]]) <- paste("lm Fit for Obs",1:noDraws)
  }
  names(simDat1List[[i]]) <- paste("Generated from",MNames)
  names(simDat2List[[i]]) <- paste("Generated from",MNames)
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
names(simDat1List) <- paste("Draws of n=",ns,sep="")
names(simDat2List) <- paste("Draws of n=",ns,sep="")
proc.time() - ptm
```

```
##    user  system elapsed
##   89.39    2.55  100.16
```

```
# create list of bases to simulate data
basisMats <- list()
for(i in 1:nsLen)
{
  basisMats[[i]] <- list()
}

for(i in 1:nsLen)
{
  for(j in 1:MLen)
  {
    basisMats[[i]][[j]] <- matrix(basisList[[i]][[j]],
                                  nrow=nrow(basisList[[i]][[j]]),
                                  ncol=ncol(basisList[[i]][[j]]))
  }
  names(basisMats[[i]]) <- paste("Basis Using",MNames)
}
names(basisMats) <- paste("Draws of n=",ns,sep="")

# simulate distributions to estimate probabilities
set.seed(1000)
ptm <- proc.time() # start timing
ICsSimDat1 <- lmCompsRcpp(simDat1List,basisMats)
```

```
## Its. for Sample Size Index 1 completed
## Its. for Sample Size Index 2 completed
## Its. for Sample Size Index 3 completed
## Its. for Sample Size Index 4 completed
## Its. for Sample Size Index 5 completed
```

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

```
##    user  system elapsed
##   46.58    0.74   72.13
```

```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("lm 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("lm Fit for Obs",1:noDraws)
  }
  names(minICList[[i]]) <- paste("Generated from",MNames)
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
## [1] 4
## [1] 5
```

```r
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("lm Fits 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
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

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

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

```
##              used   (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells    1939302  103.6    3039667  162.4   3039667  162.4
## Vcells  441289083 3366.8  703064273 5364.0 585820228 4469.5
```

```r
# ECIC step #4c
ptm <- proc.time() # start timing
DGOFList <- lmDGOFsRcpp(simDat2List,basisMats)
```

```
## Its. for Sample Size Index 1 completed
## Its. for Sample Size Index 2 completed
## Its. for Sample Size Index 3 completed
## Its. for Sample Size Index 4 completed
## Its. for Sample Size Index 5 completed
```

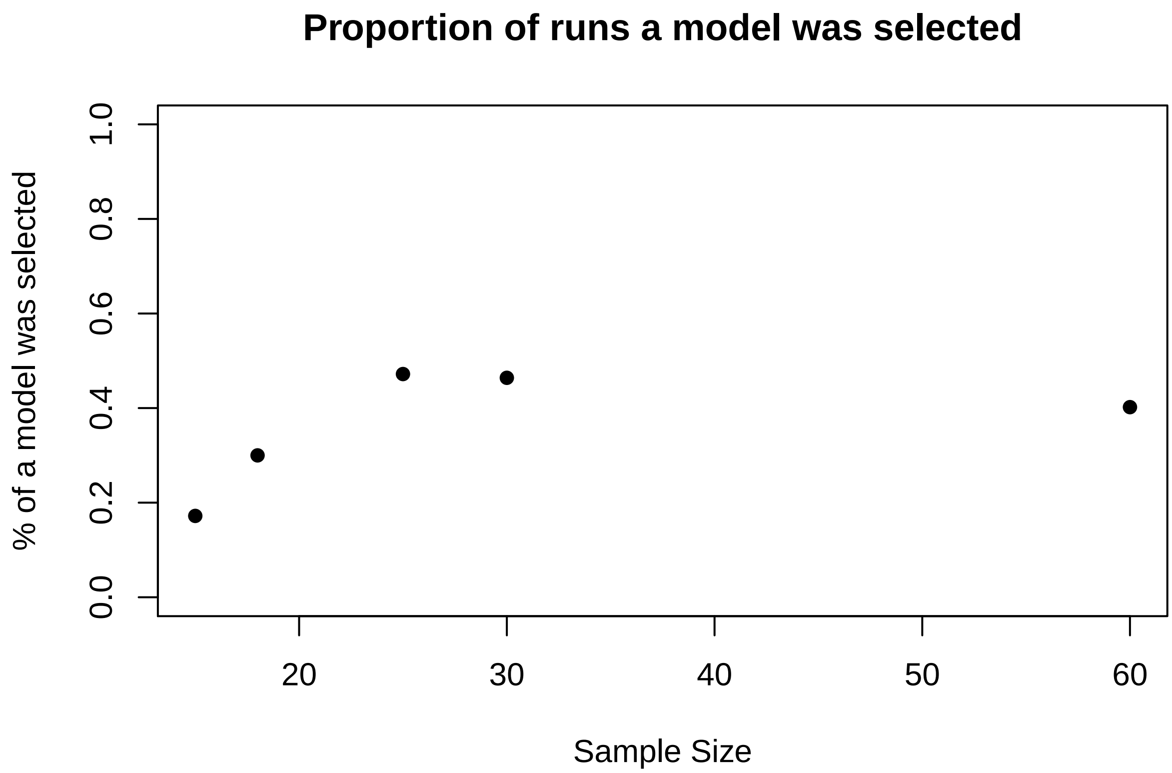
```r
proc.time() - ptm
```

```
##    user  system elapsed
##   64.32    5.89  187.48
```

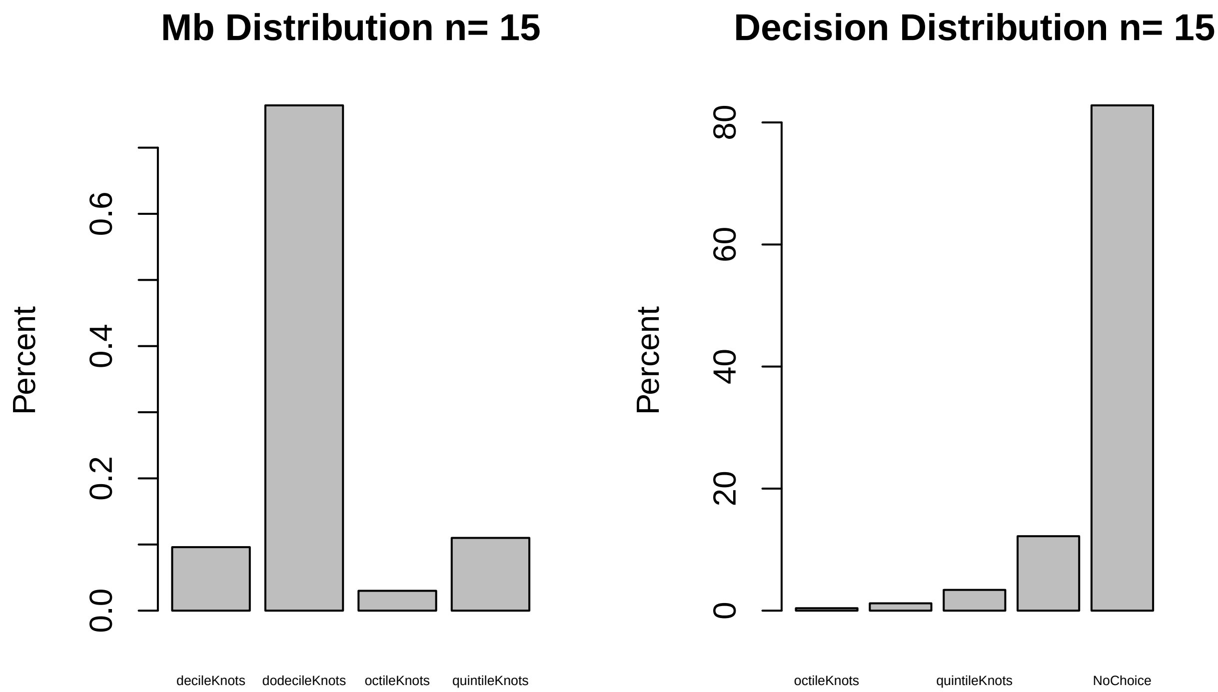
```r
# label the elements in DGOFList
for(i in 1:nsLen)
{
  for(j in 1:MLen)
  {
    for(k in 1:noDraws)
    {
      colnames(DGOFList[[i]][[j]][[k]]) <-
        paste("DGOF Under", MNames," Observed Best")
    }
    names(DGOFList[[i]][[j]]) <- paste("lm Fit for Obs",1:noDraws)
  }
  names(DGOFList[[i]])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

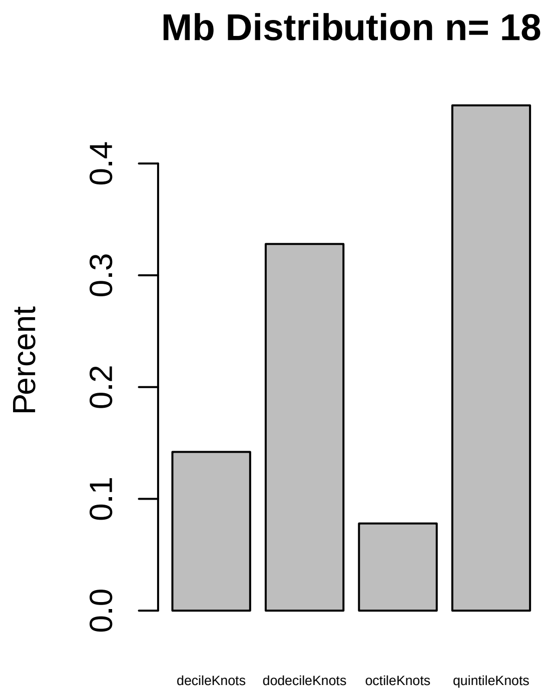
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

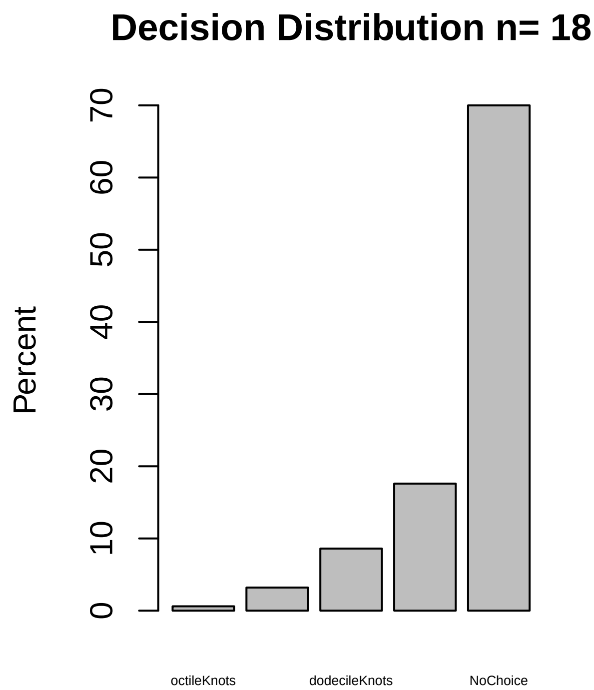
```r
######################################################################
#                      Plot Results
######################################################################

# 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))
```

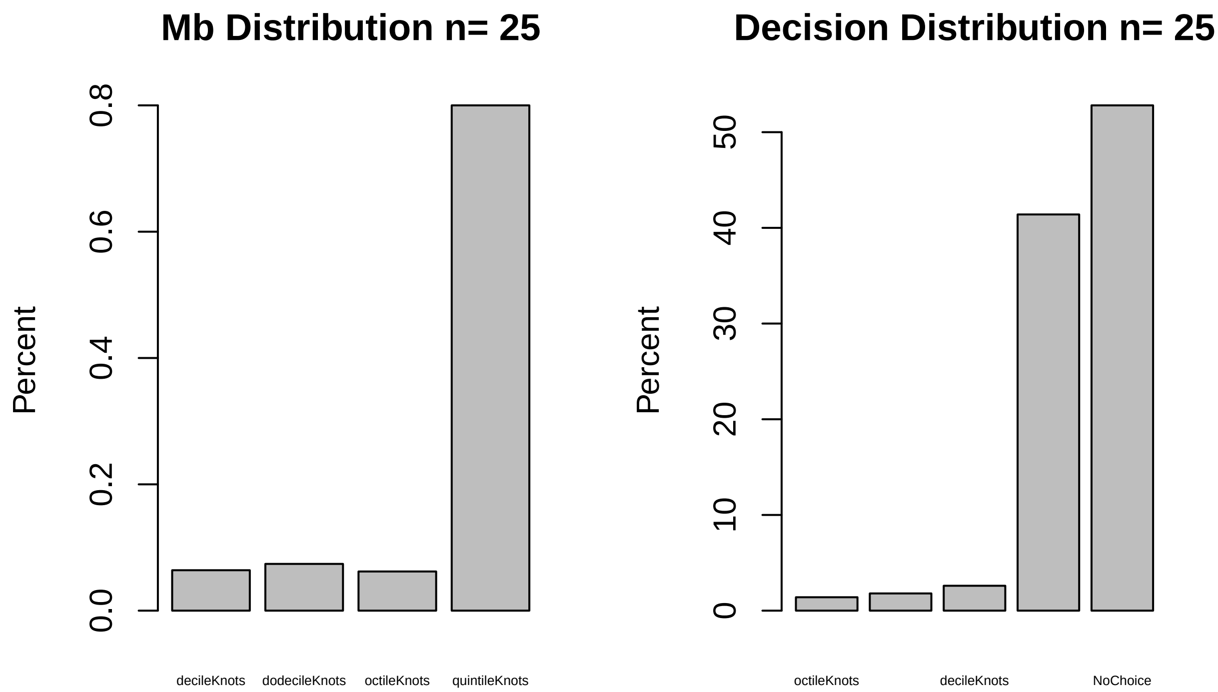


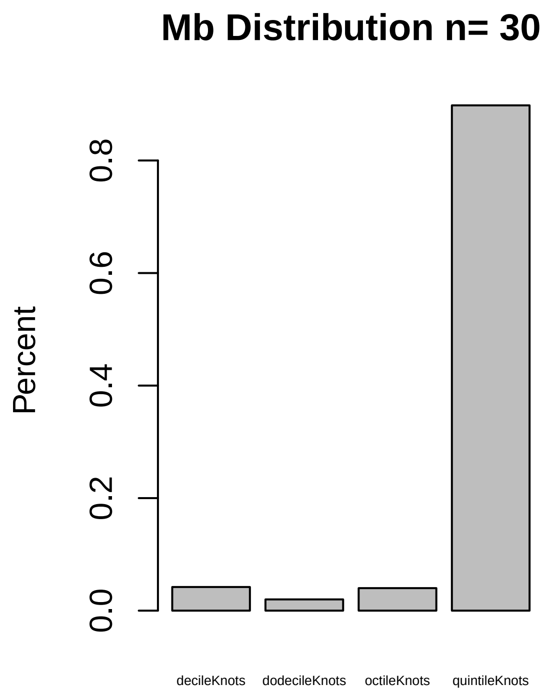

```r
# take a look at type 1 error rate
# subset assess list by only when a decision was made i.e. second row = 1
decAssessList <- lapply(X=assessList,FUN=function(x) x[,x[2,]==1])
par(mfrow = c(1, 2))
for(i in 1:nsLen)
{
  # get the frequencies of models selected as best
  tempTable <- table(MbList[[i]])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

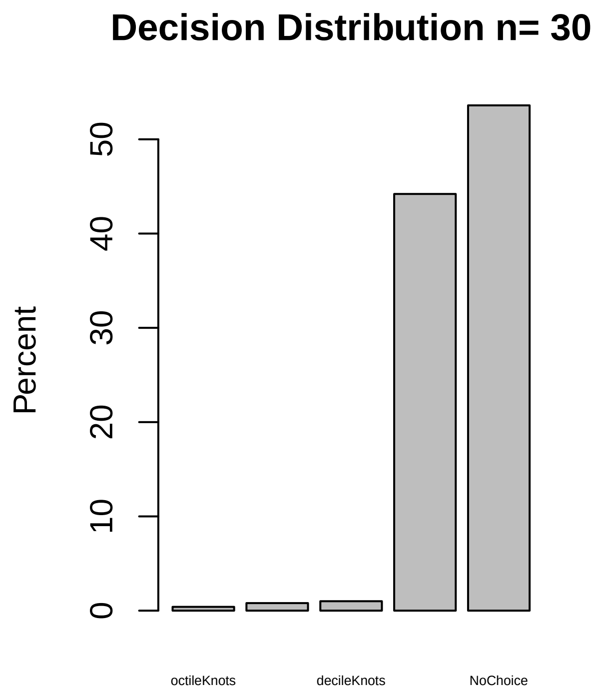

Mb Distribution n= 15
Percent
0.0
0.2
0.4
0.6
decileKnots
dodecileKnots
octileKnots
quintileKnots
Decision Distribution n= 15
Percent
0
20
40
60
80
octileKnots
quintileKnots
NoChoice

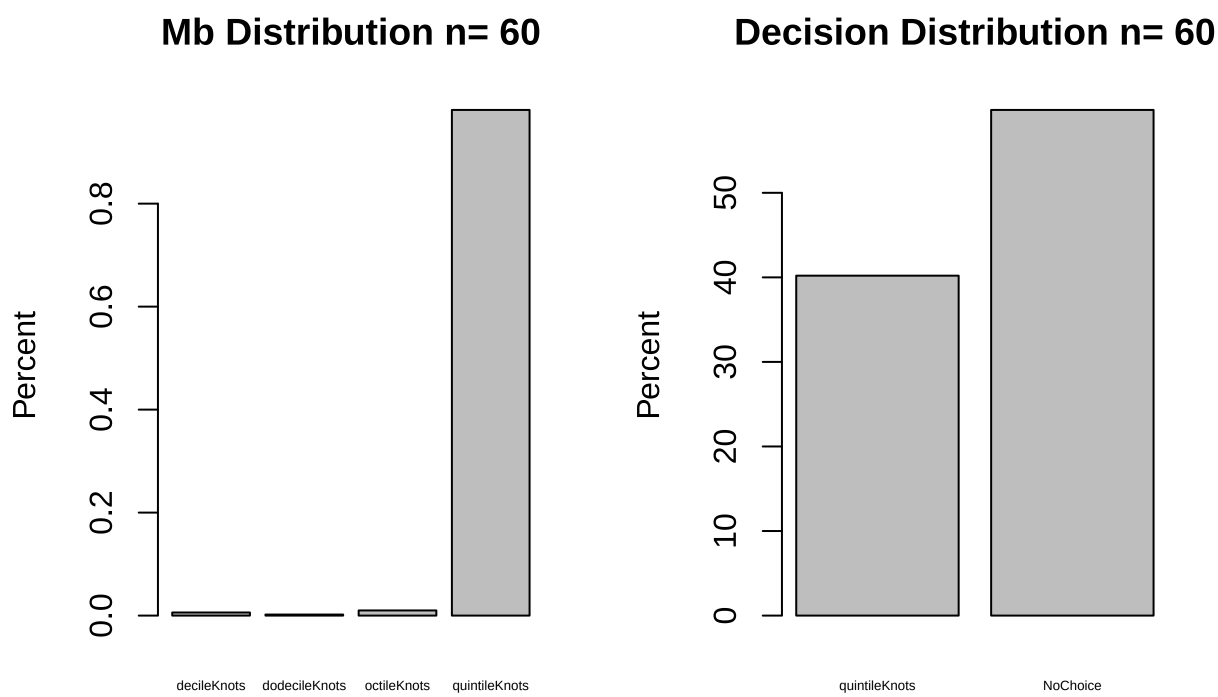

Mb Distribution n= 18
Percent
0.0
0.1
0.2
0.3
0.4
decileKnots
dodecileKnots
octileKnots
quintileKnots


Decision Distribution n= 18
Percent
0
10
20
30
40
50
60
70
octileKnots
dodecileKnots
NoChoice

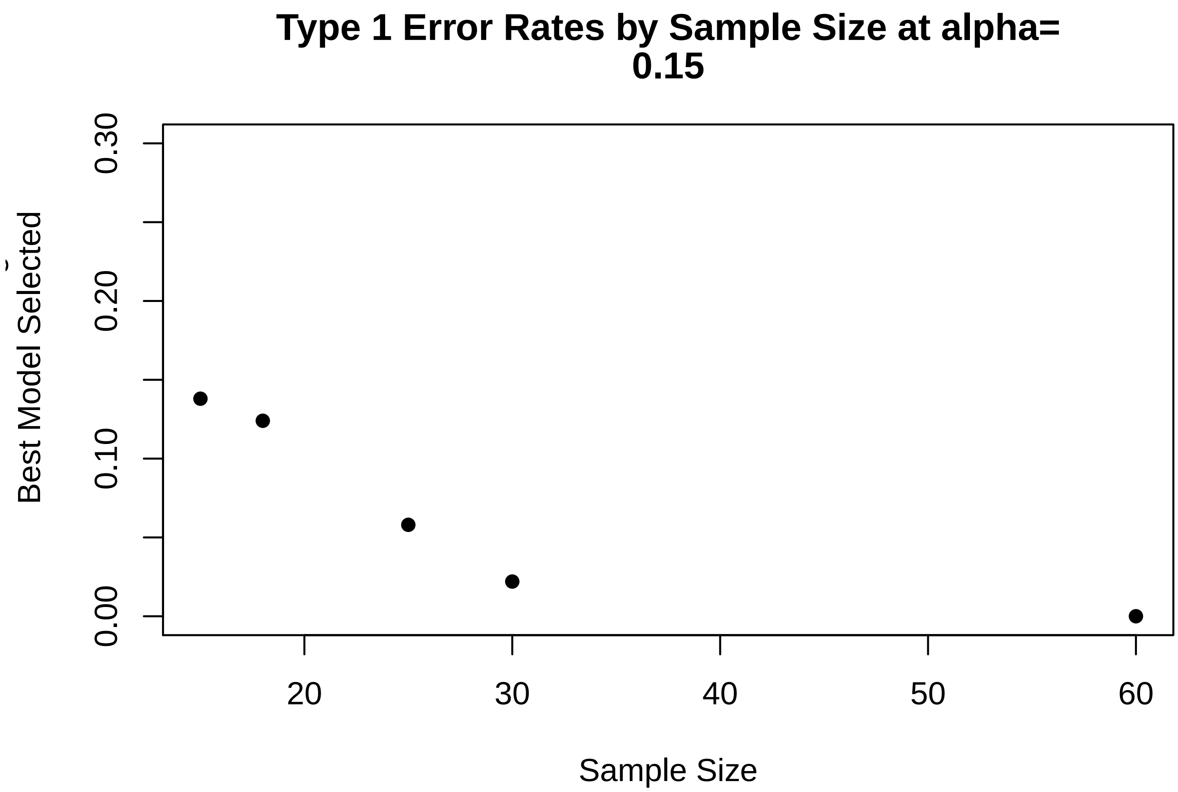

Mb Distribution n= 25
Percent
0.0
0.2
0.4
0.6
0.8
decileKnots
dodecileKnots
octileKnots
quintileKnots
Decision Distribution n= 25
Percent
0
10
20
30
40
50
octileKnots
decileKnots
NoChoice


Mb Distribution n= 30
Percent
0.0
0.2
0.4
0.6
0.8
decileKnots
dodecileKnots
octileKnots
quintileKnots

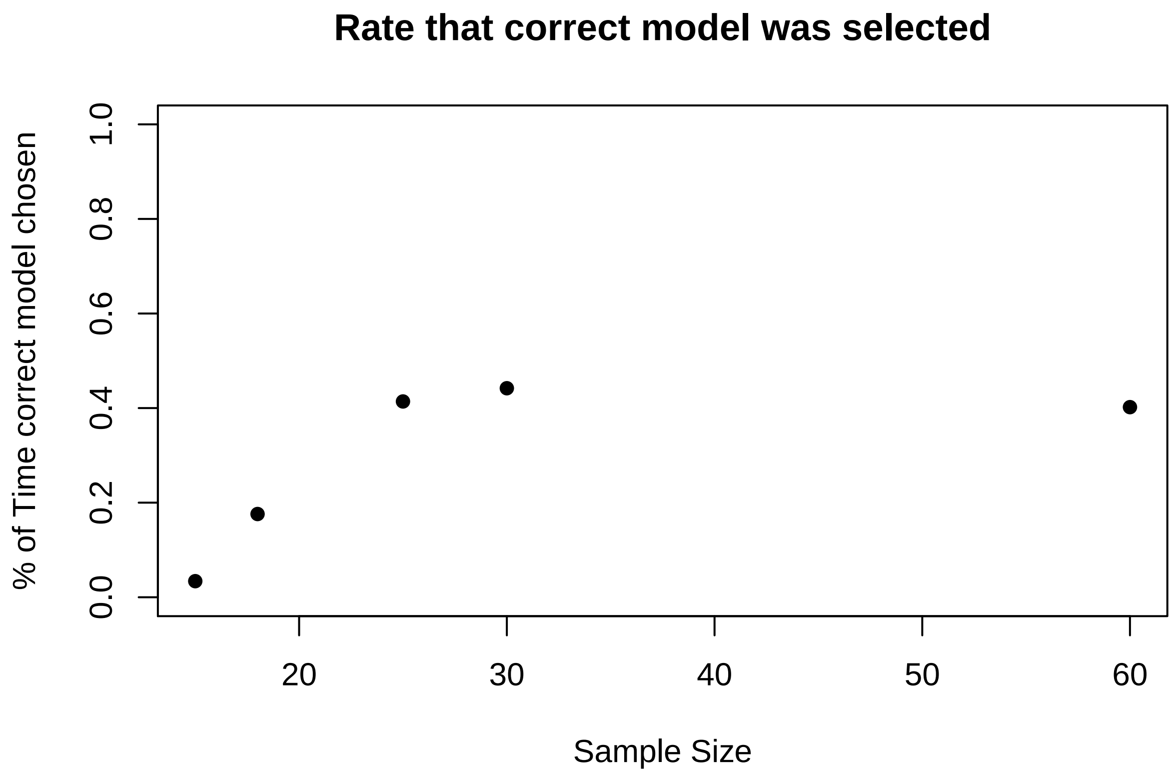

Decision Distribution n= 30
Percent
0
10
20
30
40
50
octileKnots
decileKnots
NoChoice




```r
par(mfrow = c(1, 1))
# compute type 1 error rate
t1ErrorRates <- sapply(X=decAssessList,FUN=function(x) sum(x[1,]!=trueMod)/
                         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,]==trueMod)/
                       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
20
30
40
50
60
Sample Size