Homework 1

Brett Burk

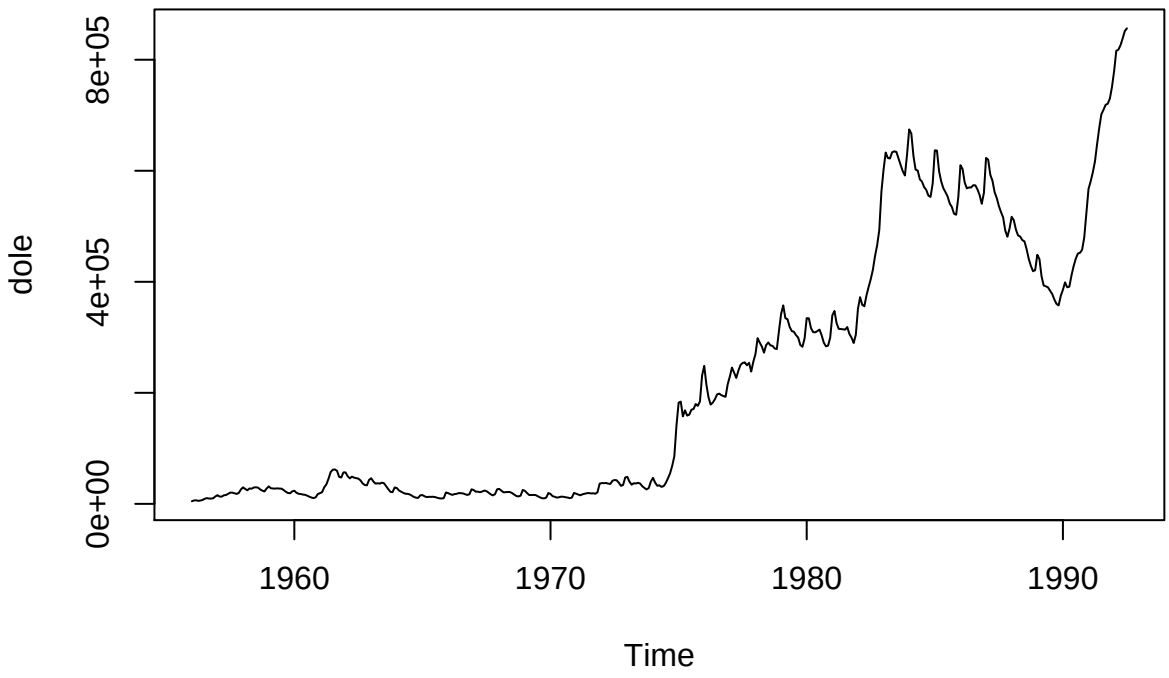
Saturday, June 13, 2015

```
library(ggplot2)
library(fma)
library(zoo)
library(mlbench)
library(caret)
```

Hyndman and Athanasopoulos

Question 2.1a objective: Analyze the unemployed benefits in Australia, and then apply a transformation if appropriate.

```
question_2.1a <- function() {
  plot(dole)
  doleLambda  <- BoxCox.lambda(dole)
  plot(BoxCox(dole, doleLambda), xlab = 'Date',
       ylab = paste('Unemployment benefits with Lambda =', round(doleLambda, digits = 2)))
}
question_2.1a()
```

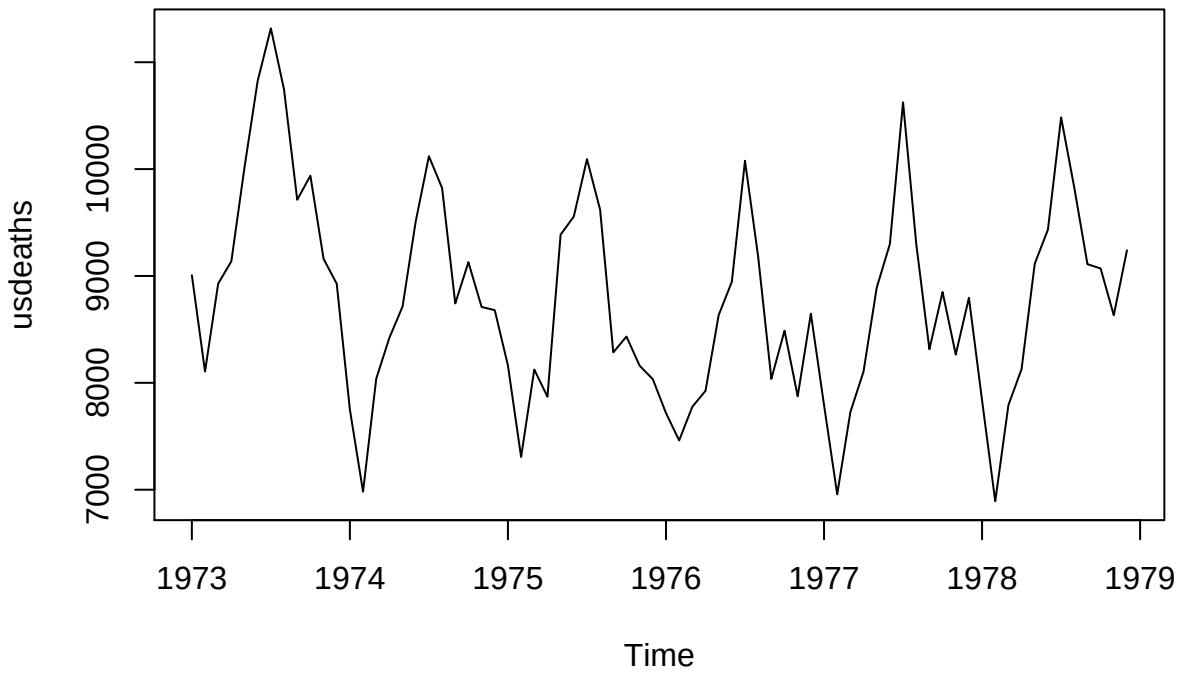

dole
0e+00
4e+05
8e+05
1960
1970
1980
1990
Time

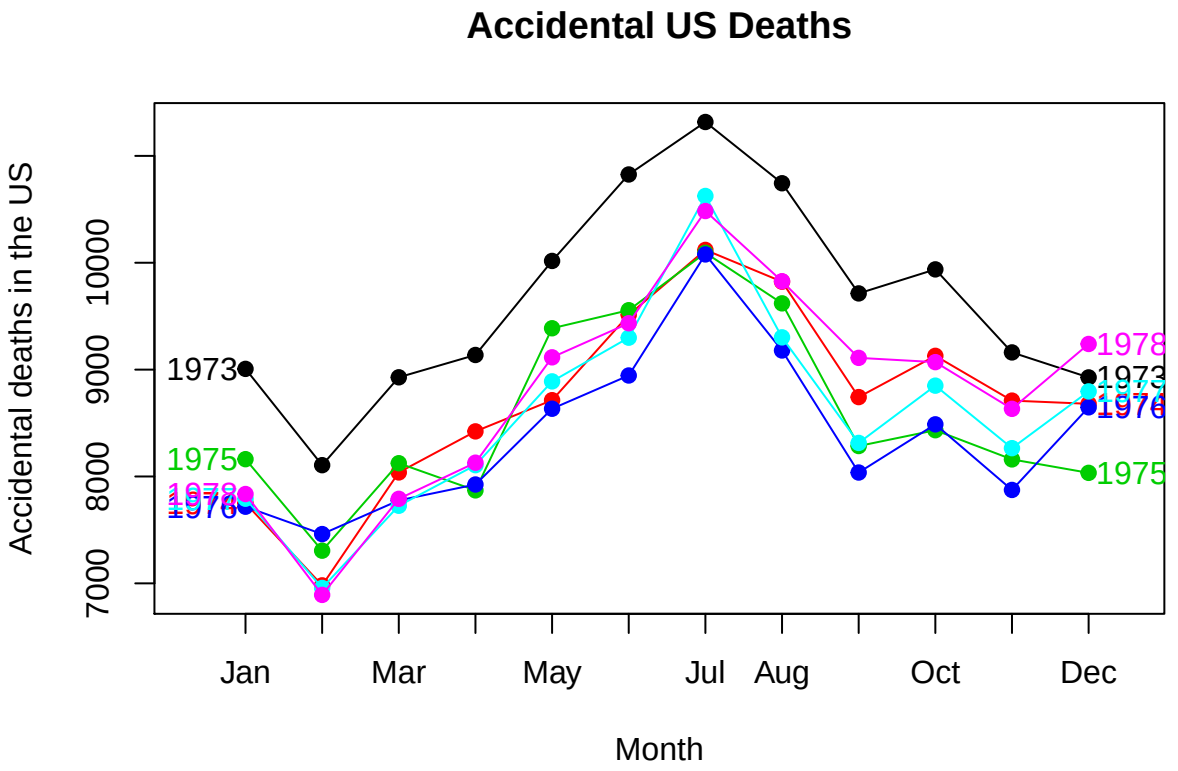

Unemployment benefits with Lambda = 0.33
50
100
150
200
250
1960
1970
1980
1990
Date

After evaluating the original plot, I calculated the lambda and then plotted a BoxCox transformation of the data–which better displays the patterns.

Question 2.1b objective: Analyze the monthly total of accidental deaths in the US, and then apply a transformation if appropriate.

```
question_2.1b <- function() {
  plot(usdeaths)
  seasonplot(usdeaths, ylab = 'Accidental deaths in the US', year.labels = TRUE,
             year.labels.left = TRUE, col = 1:6, pch = 19, main= 'Accidental US Deaths')
}
question_2.1b()
```

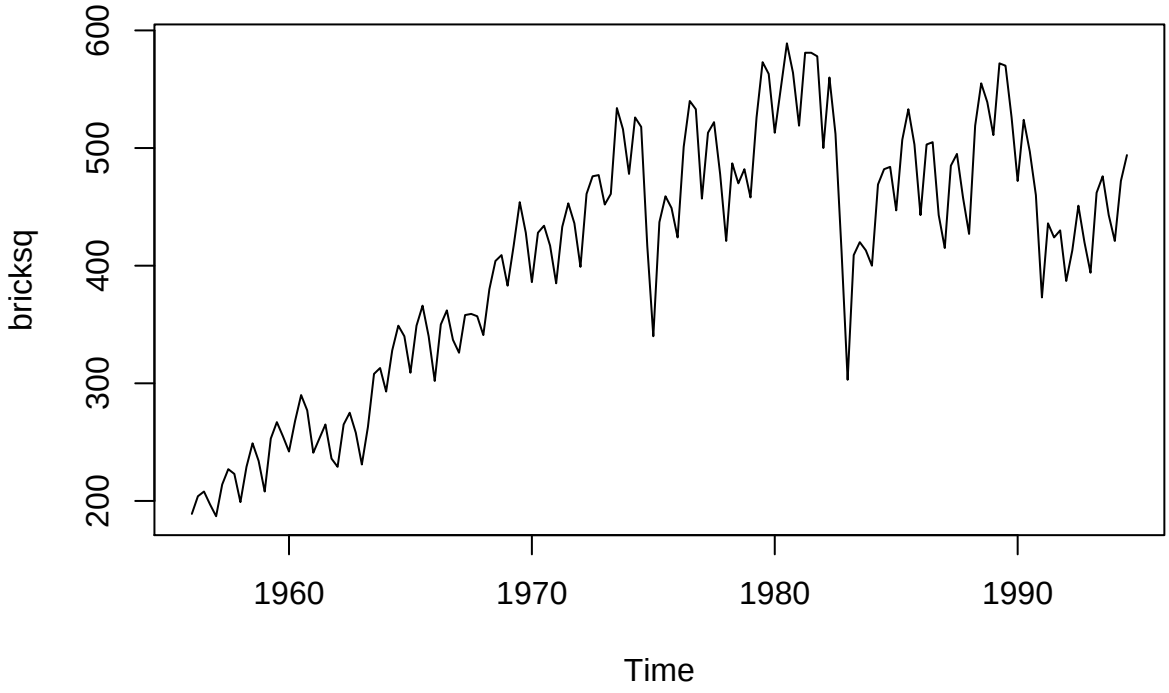

usdeaths
7000
8000
9000
10000
1973
1974
1975
1976
1977
1978
1979
Time

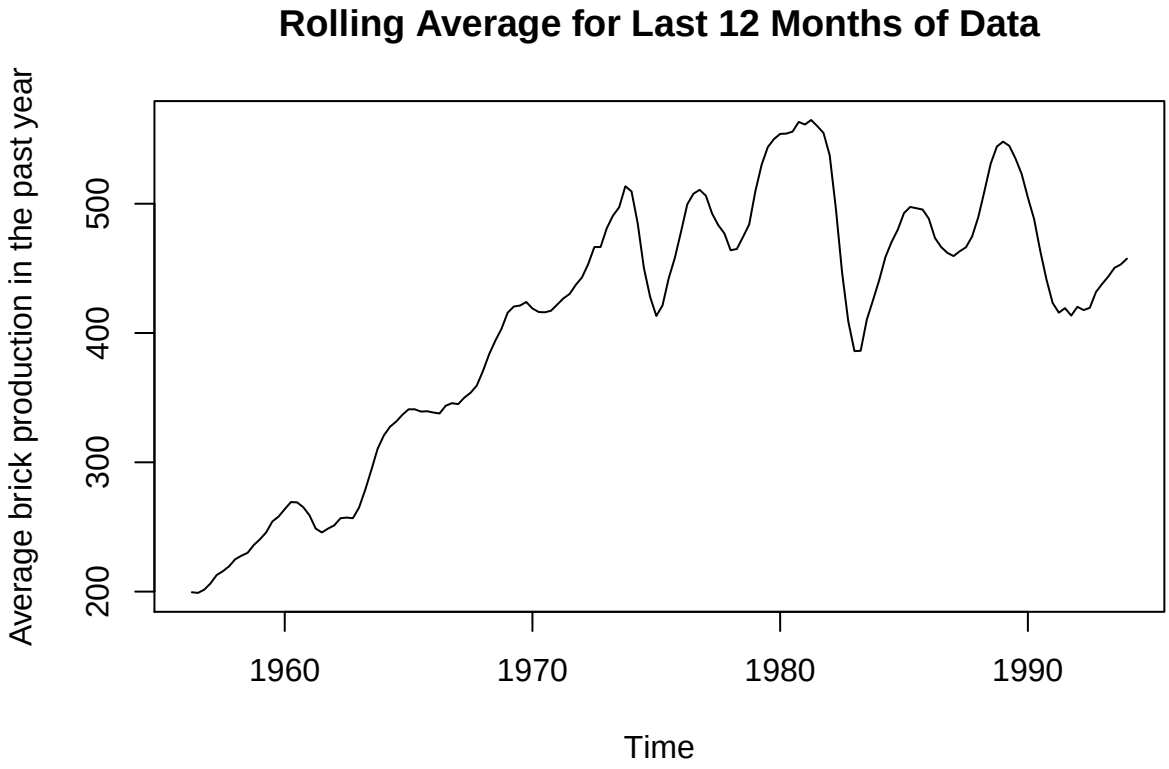

Accidental US Deaths
Accidental deaths in the US
7000
8000
9000
10000
1973
1975
1978
1973
1975
Jan
Mar
May
Jul
Aug
Oct
Dec
Month

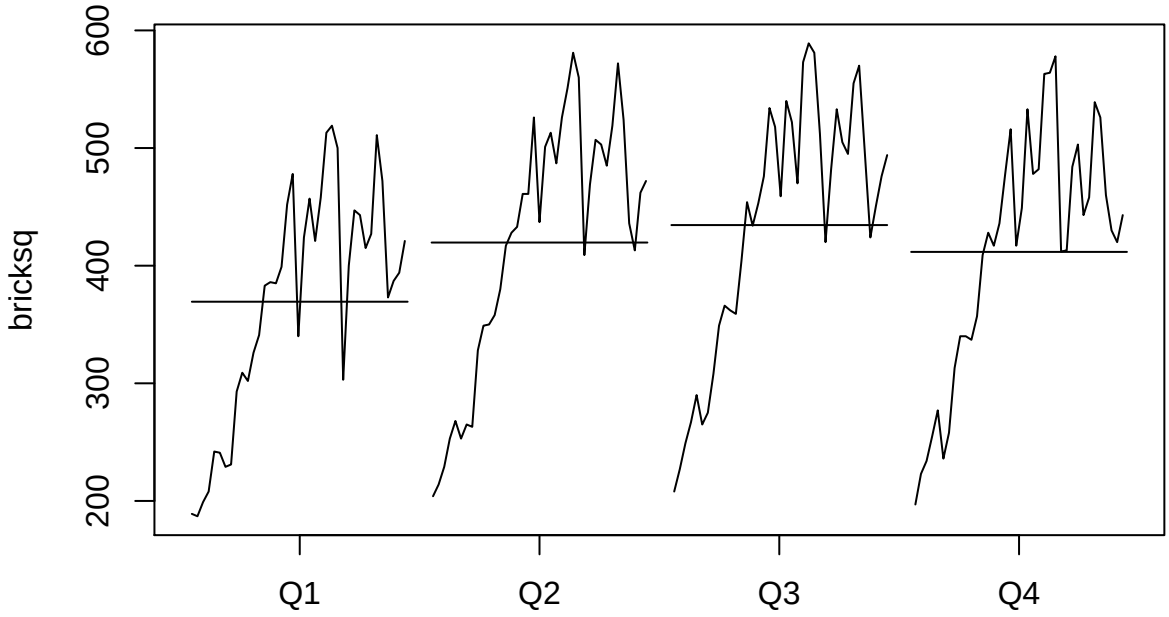
From the original plot, one can easily see that this includes seasonal data, and once it is plotted as such, we can see that the summer provides the most fatalities and also that 1973 was a year with significantly more accidental deaths than other years.

Question 2.1c objective: Analyse the quarterly total of brick production and then apply a transformation if appropriate.

```
question_2.1c <- function() {
  plot(bricksq)
  plot(rollmean(bricksq, 4), main = 'Rolling Average for Last 12 Months of Data',
       ylab = 'Average brick production in the past year')
  monthplot(bricksq)
}
question_2.1c()
```


bricksq
200
300
400
500
600
1960
1970
1980
1990
Time

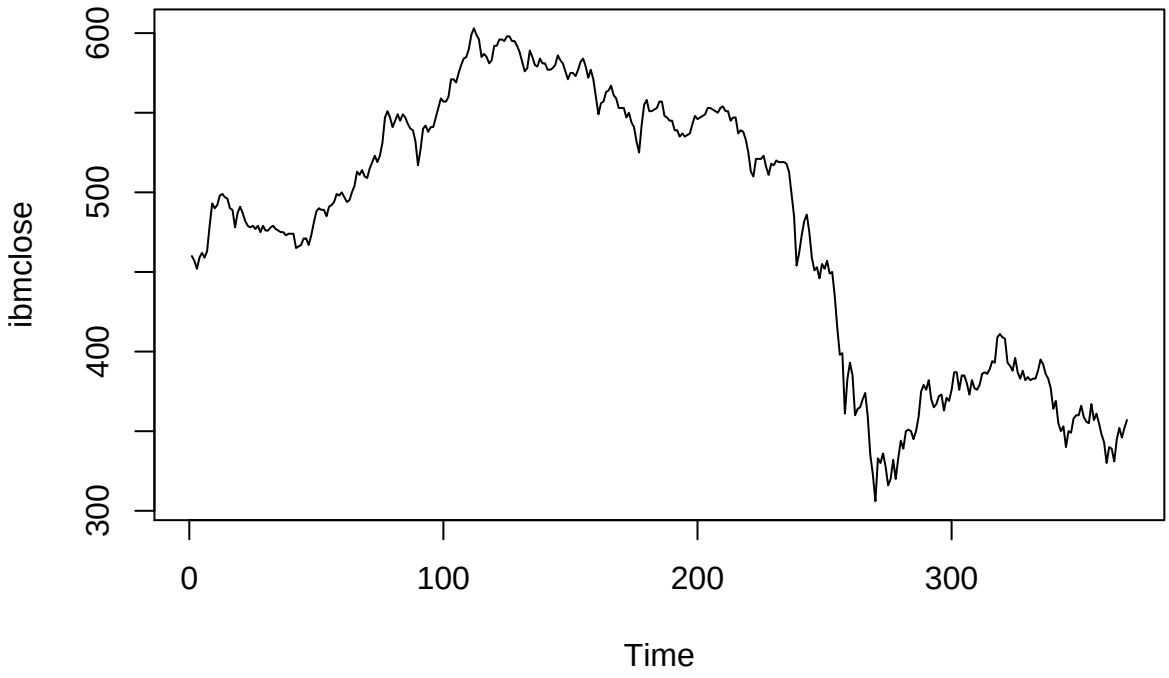

Rolling Average for Last 12 Months of Data
Average brick production in the past year
200
300
400
500
1960
1970
1980
1990
Time

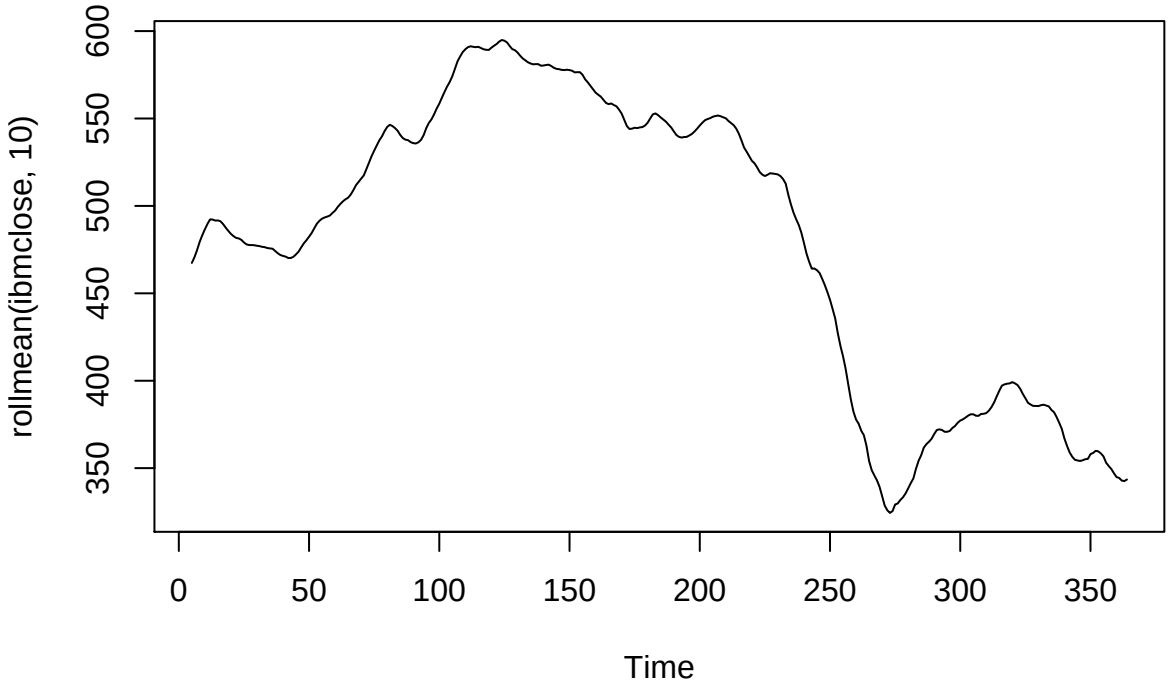


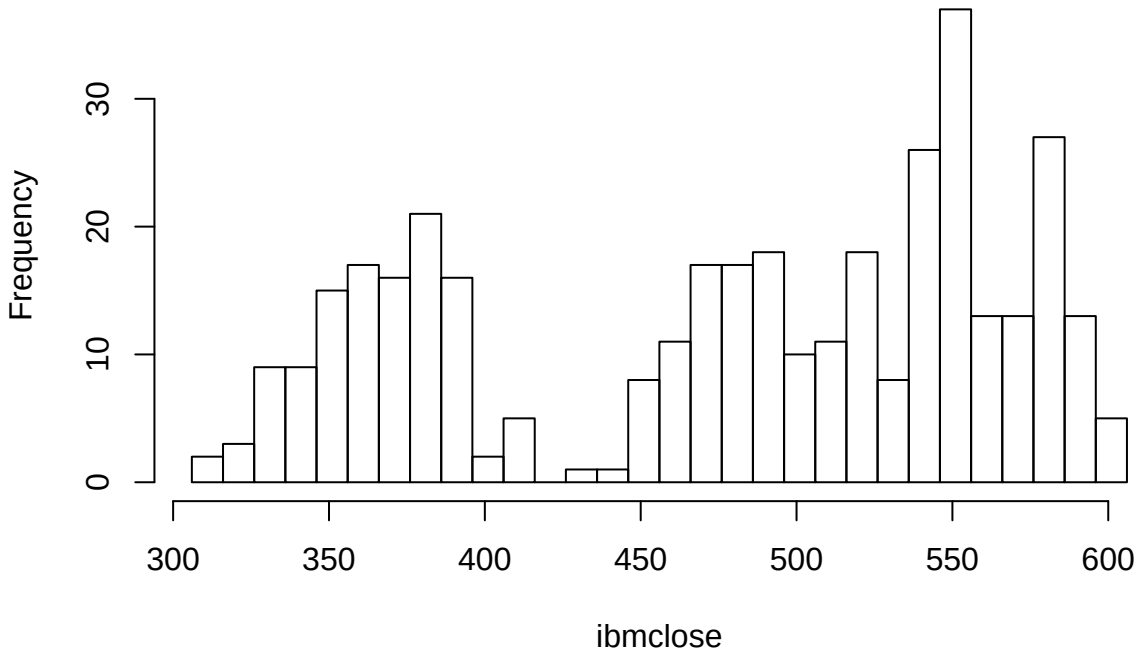
By plotting a rolling mean of the past four quarters, we can get a better idea of the overall trends (while the rolling mean was not discussed in the reading, it seemed like an appropriate transformation given the data). This also allows us to see how the past year of brick production has been for each quarter.

Question 2.3a objective: Plot and analyze the ibm data in order to become familiar with it

```
question_2.3a <- function() {
  plot(ibmclose)
  plot(rollmean(ibmclose, 10))
  hist(ibmclose, breaks = seq(min(ibmclose),
                              max(ibmclose) + 10, 10))
}
question_2.3a()
```


ibmclose
300
400
500
600
0
100
200
300
Time


rollmean(ibmclose, 10)
350
400
450
500
550
600
0
50
100
150
200
250
300
350
Time



Question 2.3b objective: Splitting the data for the first 300 and then making predictions on the alst 69 units (for training and testing purposes)

```
question_2.3b <- function() {
  trainIbm <- ibmclose[1:300]
  testIbm <- ibmclose[301:369]
}
question_2.3b()
```

Question 2.3c objective: Trying various benchmark methods (Mean, Drift, and Naive Bayes) and seeing which performs the best

```
question_2.3c <- function() {
  trainIbm <- ibmclose[1:300]
  testIbm <- ibmclose[301:369]
  ibmMean <- meanf(trainIbm, h = 10)
  ibmDrift <- rwf(trainIbm, h = 10, drift = TRUE)
  ibmNaive <- naive(trainIbm, h = 10)
  print('Mean')
  print(accuracy(ibmMean, testIbm))
  print('Drift')
  print(accuracy(ibmDrift, testIbm))
  print('Naive')
  print(accuracy(ibmNaive, testIbm))
```

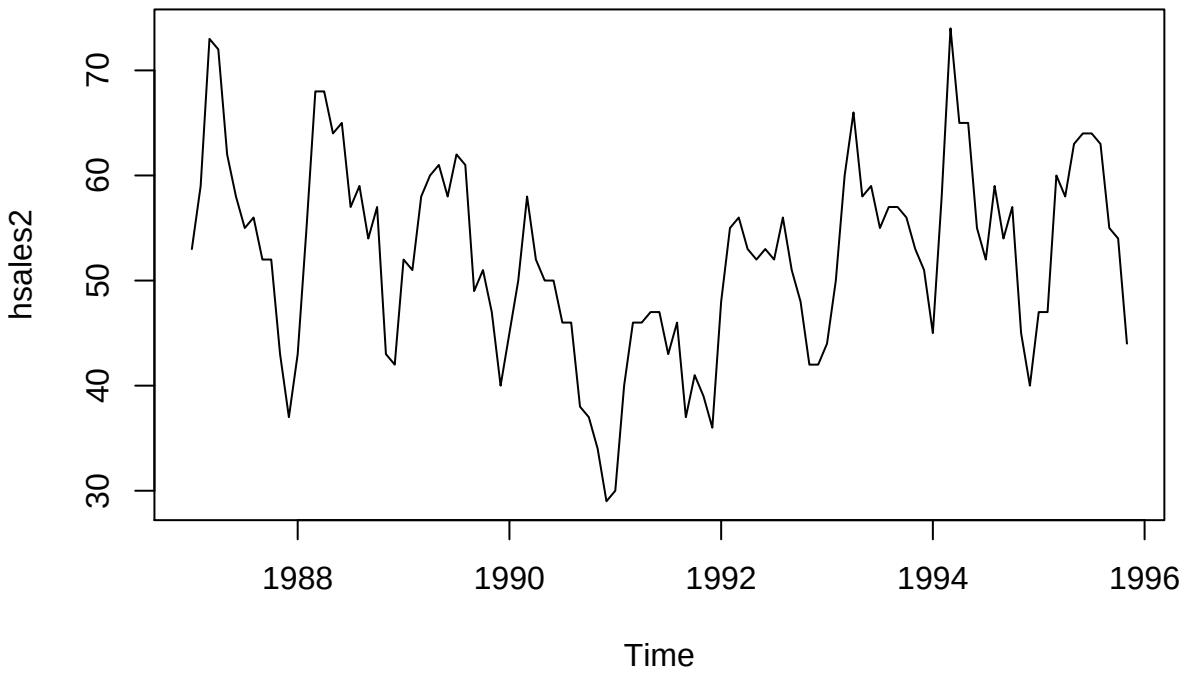
```
}
question_2.3c()
```

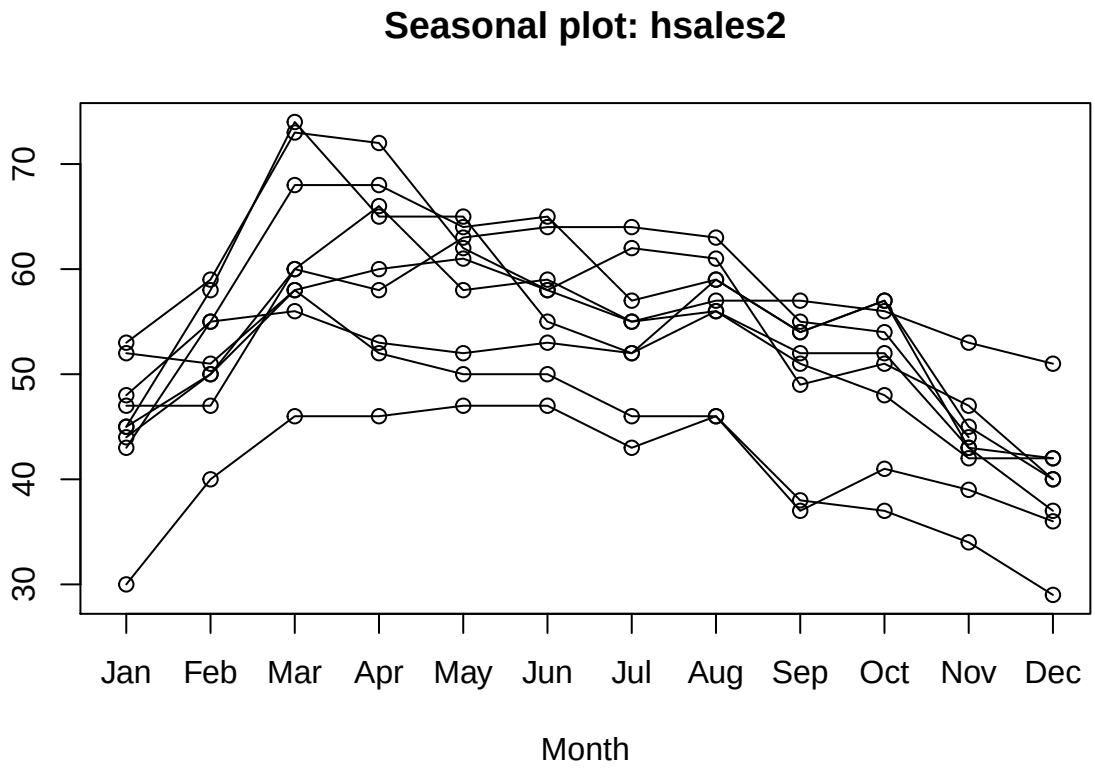
```
## [1] "Mean"
##                          ME      RMSE       MAE        MPE     MAPE MASE
## Training set  1.660438e-14  73.61532  58.72231  -2.642058 13.03019    1
## Test set     -1.220933e+02 122.18978 122.09333 -32.083817 32.08382   NA
## [1] "Drift"
##                         ME     RMSE      MAE         MPE     MAPE     MASE
## Training set 2.870480e-14 7.297409 5.127996 -0.02530123 1.121650 1.006083
## Test set     6.345151e+00 7.708126 6.551839  1.65200211 1.707415       NA
##                   ACF1
## Training set 0.1351052
## Test set            NA
## [1] "Naive"
##                        ME     RMSE     MAE         MPE     MAPE MASE
## Training set -0.2809365 7.302815 5.09699 -0.08262872 1.115844    1
## Test set      4.8000000 6.826419 5.40000  1.24443538 1.405293   NA
##                   ACF1
## Training set 0.1351052
## Test set            NA
```

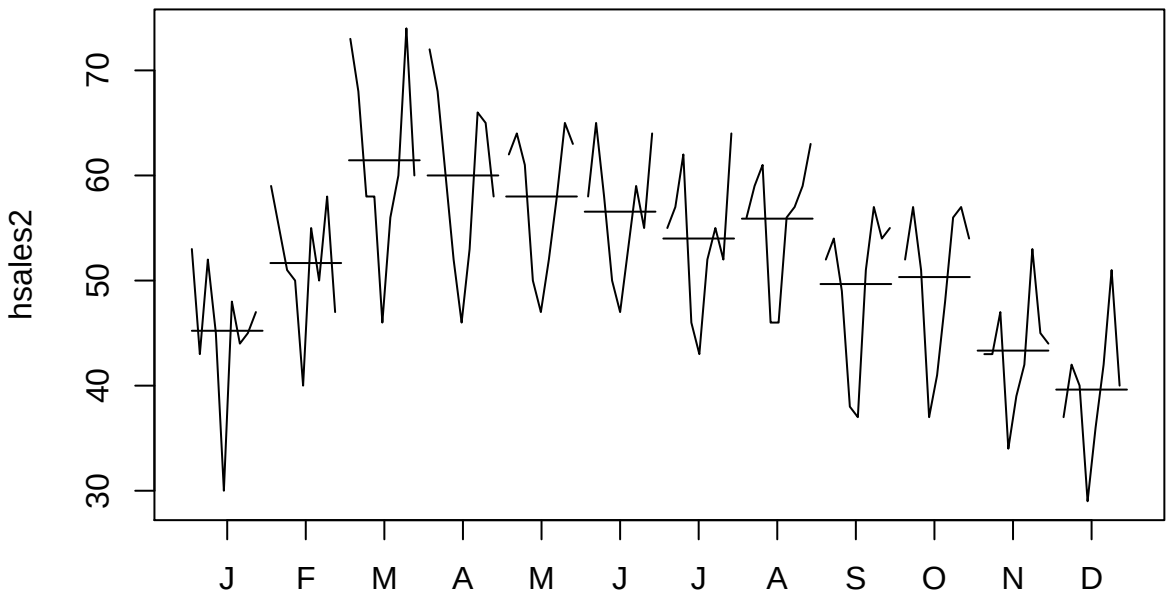
As it had the lowest RMSE, MAE, MPE, and MAPE for the test set, the Naive Bayes method worked the best

Question 2.4a objective: Plotting the one family houses by first a time series, then a season plot, and then a month plot in order to interpret the results

```
question_2.4a <- function() {
  plot(hsales2)
  seasonplot(hsales2)
  monthplot(hsales2)
}
question_2.4a()
```


hsales2
30
40
50
60
70
1988
1990
1992
1994
1996
Time


Seasonal plot: hsales2
30
40
50
60
70
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month



We can see that purchases are typically at the ends of the month, and also that March and April typically see more purchases.

Question 2.4b objective: Seperating the last two years worth of data in order to test predictions on it

```
question_2.4b <- function() {
  salesTrain <- window(hsales2, end = 1994 - 0.1)
  salesTest <- window(hsales2, start = 1994 - 0.1)
}
question_2.4b()
```

Question 2.4c objective: Testing the accuracy of various predictive models

```
question_2.4c <- function() {
  salesTrain <- window(hsales2, end = 1994 - 0.1)
  salesTest <- window(hsales2, start = 1994 - 0.1)
  salesMean <- meanf(salesTrain, h = 10)
  salesDrift <- rwf(salesTrain, h = 10, drift = TRUE)
  salesNaive <- naive(salesTrain, h = 10)
  salesSNaive <- snaive(salesTrain, h = 10)
  print('Mean')
  print(accuracy(salesMean, salesTest))
  print('Drift')
  print(accuracy(salesDrift, salesTest))
```

```
  print('Naive')
  print(accuracy(salesNaive, salesTest))
  print('Seasonal Naive')
  print(accuracy(salesSNaive, salesTest))
}
question_2.4c()
```

```
## [1] "Mean"
##                          ME      RMSE      MAE       MPE     MAPE     MASE
## Training set -2.995476e-15  9.139601 7.334882 -3.565894 15.46723 1.096372
## Test set      6.559036e+00 10.314114 7.855422  9.695770 12.56403 1.174179
##                   ACF1 Theil's U
## Training set 0.8080110        NA
## Test set     0.3799874  1.115619
## [1] "Drift"
##                ME     RMSE      MAE        MPE      MAPE      MASE
## Training set 0.0 5.691243 4.365854 -0.6630402  8.799516 0.6525802
## Test set     4.8 9.295160 7.000000  6.5957429 11.320228 1.0463158
##                   ACF1 Theil's U
## Training set 0.1524849        NA
## Test set     0.3799874  1.006572
## [1] "Naive"
##                         ME     RMSE      MAE        MPE      MAPE      MASE
## Training set 3.151518e-13 5.691243 4.365854 -0.6630402  8.799516 0.6525802
## Test set     4.800000e+00 9.295160 7.000000  6.5957429 11.320228 1.0463158
##                   ACF1 Theil's U
## Training set 0.1524849        NA
## Test set     0.3799874  1.006572
## [1] "Seasonal Naive"
##                        ME     RMSE      MAE       MPE      MAPE      MASE
## Training set -0.2112676 7.939028 6.690141 -1.718050 14.143562 1.0000000
## Test set      3.0000000 6.557439 5.200000  4.660439  8.687634 0.7772632
##                       ACF1 Theil's U
## Training set  0.774978995        NA
## Test set     -0.002941176 0.6855548
```

Seasonal Naive Bayes performs the best.

Kuhn and Johnson

Question 3.2a objective: Examine the data and identify degenerate cases

```
question_3.2a <- function() {
  data(Soybean)
  for(name in names(Soybean)){
    print(name)
    print(table(Soybean[name]))
    print('------------------------------')
  }
}
question_3.2a()
```

```
## [1] "Class"
##
##               2-4-d-injury         alternarialeaf-spot
##                         16                          91
##                anthracnose            bacterial-blight
##                         44                          20
##          bacterial-pustule                 brown-spot
##                         20                          92
##             brown-stem-rot               charcoal-rot
##                         44                          20
##              cyst-nematode diaporthe-pod-&-stem-blight
##                         14                          15
##       diaporthe-stem-canker               downy-mildew
##                         20                          20
##          frog-eye-leaf-spot            herbicide-injury
##                         91                           8
##      phyllosticta-leaf-spot            phytophthora-rot
##                         20                          88
##              powdery-mildew           purple-seed-stain
##                         20                          20
##        rhizoctonia-root-rot
##                         20
## [1] "------------------------------"
## [1] "date"
##
##   0   1   2   3   4   5   6
##  26  75  93 118 131 149  90
## [1] "------------------------------"
## [1] "plant.stand"
##
##   0   1
## 354 293
## [1] "------------------------------"
## [1] "precip"
##
##   0   1   2
##  74 112 459
## [1] "------------------------------"
## [1] "temp"
##
##   0   1   2
##  80 374 199
## [1] "------------------------------"
## [1] "hail"
##
##   0   1
## 435 127
## [1] "------------------------------"
## [1] "crop.hist"
##
##   0   1   2   3
##  65 165 219 218
## [1] "------------------------------"
## [1] "area.dam"
```

```
##
##   0   1   2   3
## 123 227 145 187
## [1] "------------------------------"
## [1] "sever"
##
##   0   1   2
## 195 322  45
## [1] "------------------------------"
## [1] "seed.tmt"
##
##   0   1   2
## 305 222  35
## [1] "------------------------------"
## [1] "germ"
##
##   0   1   2
## 165 213 193
## [1] "------------------------------"
## [1] "plant.growth"
##
##   0   1
## 441 226
## [1] "------------------------------"
## [1] "leaves"
##
##   0   1
##  77 606
## [1] "------------------------------"
## [1] "leaf.halo"
##
##   0   1   2
## 221  36 342
## [1] "------------------------------"
## [1] "leaf.marg"
##
##   0   1   2
## 357  21 221
## [1] "------------------------------"
## [1] "leaf.size"
##
##   0   1   2
##  51 327 221
## [1] "------------------------------"
## [1] "leaf.shread"
##
##   0   1
## 487  96
## [1] "------------------------------"
## [1] "leaf.malf"
##
##   0   1
## 554  45
## [1] "------------------------------"
```

```
## [1] "leaf.mild"
##
##   0   1   2
## 535  20  20
## [1] "------------------------------"
## [1] "stem"
##
##   0   1
## 296 371
## [1] "------------------------------"
## [1] "lodging"
##
##   0   1
## 520  42
## [1] "------------------------------"
## [1] "stem.cankers"
##
##   0   1   2   3
## 379  39  36 191
## [1] "------------------------------"
## [1] "canker.lesion"
##
##   0   1   2   3
## 320  83 177  65
## [1] "------------------------------"
## [1] "fruiting.bodies"
##
##   0   1
## 473 104
## [1] "------------------------------"
## [1] "ext.decay"
##
##   0   1   2
## 497 135  13
## [1] "------------------------------"
## [1] "mycelium"
##
##   0   1
## 639   6
## [1] "------------------------------"
## [1] "int.discolor"
##
##   0   1   2
## 581  44  20
## [1] "------------------------------"
## [1] "sclerotia"
##
##   0   1
## 625  20
## [1] "------------------------------"
## [1] "fruit.pods"
##
##   0   1   2   3
## 407 130  14  48
```

```
## [1] "------------------------------"
## [1] "fruit.spots"
##
##   0   1   2   4
## 345  75  57 100
## [1] "------------------------------"
## [1] "seed"
##
##   0   1
## 476 115
## [1] "------------------------------"
## [1] "mold.growth"
##
##   0   1
## 524  67
## [1] "------------------------------"
## [1] "seed.discolor"
##
##   0   1
## 513  64
## [1] "------------------------------"
## [1] "seed.size"
##
##   0   1
## 532  59
## [1] "------------------------------"
## [1] "shriveling"
##
##   0   1
## 539  38
## [1] "------------------------------"
## [1] "roots"
##
##   0   1   2
## 551  86  15
## [1] "------------------------------"
```

There are quite a few degenerate cases:

- leaf.malf–1
- roots–2
- fruit.pods–2
- sclerotia–1
- mycelium–1
- Various classes, especially 'herbicide injury' and '2-4-d-injury'

Question 3.2b objective: identify patterns in missing data

```
question_3.2b <- function() {
  missingData <- Soybean[!(complete.cases(Soybean)),]
  table(missingData$Class)[table(missingData$Class) >= 1]
}
question_3.2b()
```

```
##
##                2-4-d-injury                cyst-nematode
##                         16                           14
## diaporthe-pod-&-stem-blight           herbicide-injury
##                         15                            8
##           phytophthora-rot
##                         68
```

Almost all of the missing data comes from a small collection of classes. Almost all of those classes contain only rows with missing data (phytophthora-rot is the only one that contains missing information for some but not all of its entries). Almost all of the data with missing entries have a seed size of 1. The classes with missing data: 2-4-d-injury * cyst-nematode * diaporthe-pod-&-stem-blight * herbicide-injury * phytophthora-rot

Question 3.2c objective: Strategy for handling missing data

A simple way to handle this missing data, would be to train an algorithm to make an initial fork–as the fact that information has missing values is clearly predictive in this case–if it includes missing data, than it is far more likely to be one of the initial groups. In order to enhance the predictive power, it may be easier (depending on how important it is to predict thsoe specific diseases) to simply clean the dataset by removing all non-complete cases.

```
question_3.2c <- function() {
  mungedData <- Soybean[complete.cases(Soybean),]
}
question_3.2c()
```

Question 4.4a objective: Compare the normal probabilities to those of a random sample

```
question_4.4a <- function() {
  data(oil)
  propTable <- table(oilType)/length(oilType)
  simData <- propTable
  for(i in 1:100){
    sampleItems <- sample(1:length(oilType), 60)
    sampledOil <- oilType[sampleItems]
    sampTable <- table(sampledOil)/length(sampledOil)
    simData <- rbind(simData, propTable - sampTable)
  }
  print(colMeans(simData))
}
question_4.4a()
```

```
##             A             B             C             D             E
##  1.115924e-02 -1.938944e-03  2.268977e-04  3.094059e-04  6.188119e-05
##             F             G
##  8.663366e-04 -7.838284e-04
```

Question 4.4b objective: Compare random samples with those produced by create data partition

```
question_4.4b <- function() {
  propTable <- table(oilType)/length(oilType)
  dp <- createDataPartition(oilType, p = 0.59)
  dpOil <- oilType[dp[[1]]]
  dpTable <- table(dpOil)/length(dpOil)
  print(propTable - dpTable)
}
question_4.4b()
```

```
## oilType
##            A            B            C            D            E
##  0.018750000  0.004166667 -0.002083333 -0.010416667 -0.002083333
##            F            G
##  0.004166667 -0.012500000
```

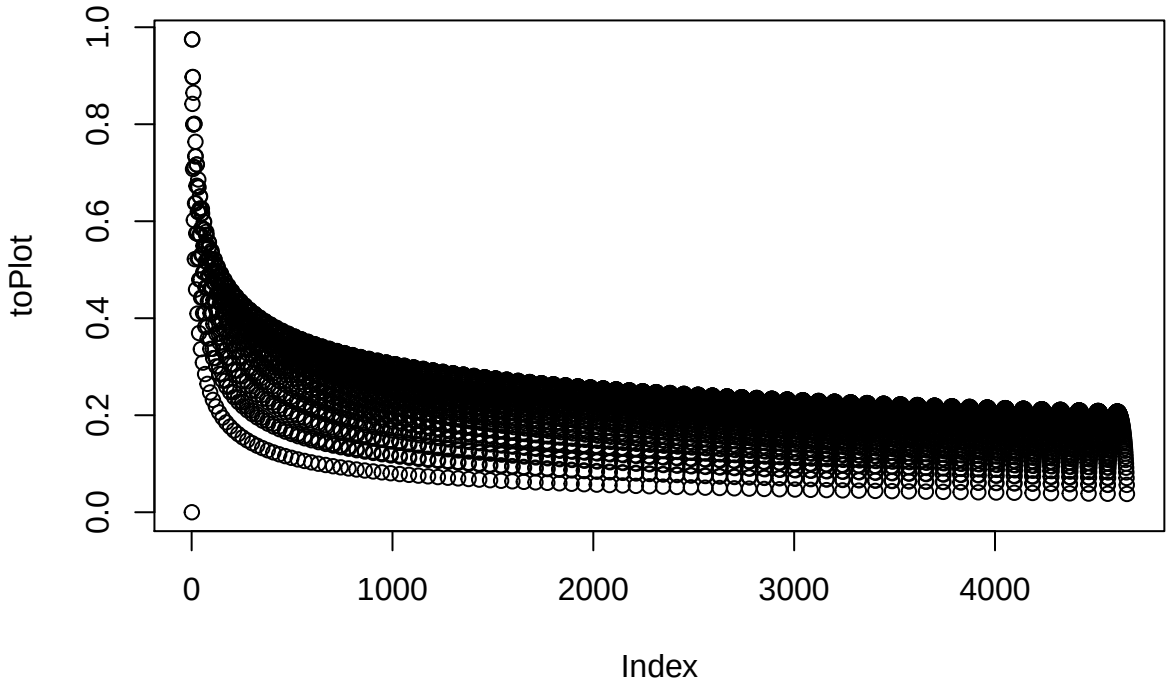
The main difference for this is that the results are uniform (it produces exactly the same result, whereas the sample is random). It attempts to build a diversified sample, which using sample() does not always produce.

Question 4.4c objective: evaluate whether creating a test/train group is efficient given the data size

Using a test model would be very inefficient, as it would require excluding too much of the data. Using an analysis that requires 'folds' or iterates through several times while leaving out portions of the data each time would be much more efficient for such a small dataset.

Question 4.4d objective: find different sizes for confidence intervals

```
question_4.4d <- function() {
  toPlot <- c(0)
  for(i in 1:length(oilType)){
    for(j in 1:i){
      currTest <- binom.test(j, i)$conf.int
      toPlot[length(toPlot)+1] <- currTest[2] - currTest[1]
    }
  }
  plot(toPlot)
}
question_4.4d()
```



This plot is far from perfect, but it does allow us to see the important information–i.e. as the size of the sample gets closer to the size of the data set, and the amount of correct answers gets larger, our confidence interval gets progressively smaller. This makes intuitive sense, and the plot, while simple, confirms this.