Project 1

John Macke, Pujan Thakrar, Mark Vandre

October 29, 2018

Question 1: Regression - Least Squares, Bootstrap, and Bayesian

For this problem we will compare regression estimates between Least Squares (LS), Bayesian, and Bootstrapping methods. Using any dataset of your choice from the AER library (you will need to install the package in R first), estimate a multiple linear regression model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + e$. Your data will need to have at least 2 predictors, and the LS estimates of the parameters should be statistically significant at the 5% level.

We chose the CASchools dataset from the AER package which contains data on test performance, school characteristics and student demographic backgrounds for school districts in California.

We then created a “student-teacher ratio” variable as well as a “average test score” variable using data provided.

We then ran a simple linear regression where we regress average test scores on student-teacher ratio, percentage qualifying for reduced-price lunch, and the district’s average income.

```
data("CASchools")                                          #load in data

CASchools$stratio <- with(CASchools,students/teachers)     #create student-teacher ratio
CASchools$avgscore<-with(CASchools,(read+math)/2)          #create average score

fm <-lm(avgscore~stratio+income+lunch, data = CASchools)   #run regression

summary(fm)
```

```
##
## Call:
## lm(formula = avgscore ~ stratio + income + lunch, data = CASchools)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.362  -5.222  -0.367   5.518  33.706
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 685.03118    5.30955 129.019  < 2e-16 ***
## stratio      -0.83073    0.23426  -3.546 0.000435 ***
## income        0.52727    0.08335   6.326 6.51e-10 ***
## lunch        -0.50631    0.02180 -23.226  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.819 on 416 degrees of freedom
## Multiple R-squared:  0.7873, Adjusted R-squared:  0.7858
## F-statistic: 513.3 on 3 and 416 DF,  p-value: < 2.2e-16
```

From the summary we can see that all the LS estimates of the parameters are statistically significant at the 5% level.

We then stored the estimates as well as their respective 95% confidence intervals to refer back to later.

```
ls=coef(fm)
con1=confint(fm,'(Intercept)',level=0.95)
con2=confint(fm,'stratio',level=0.95)
con3=confint(fm,'income',level=0.95)
con4=confint(fm,'lunch',level=0.95)
```

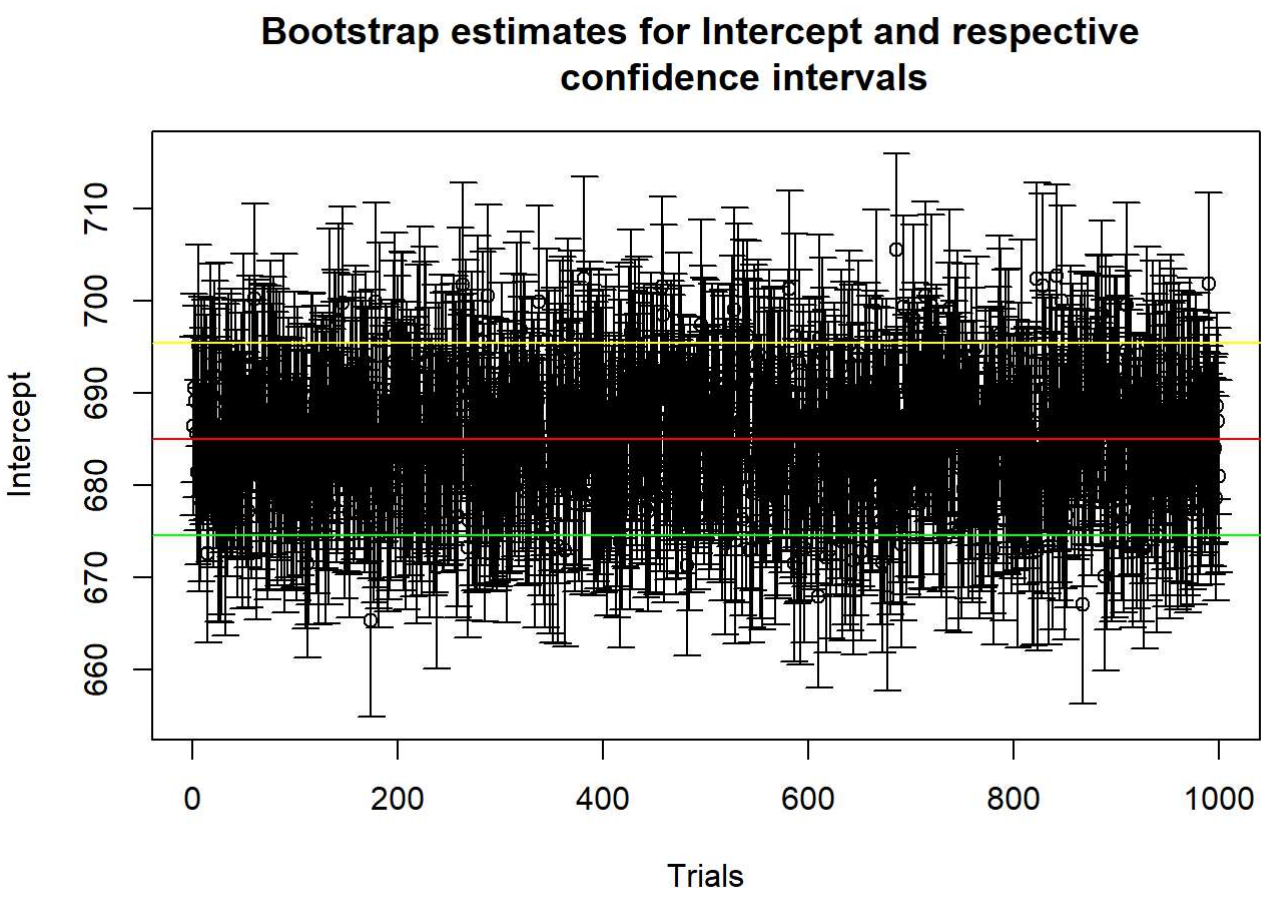
(a) Use bootstrapping with N=1000 samples to estimate the Boostrap estimates of the model parameters (β_0,β_1, and β_2), and also the respective confidence intervals. Plot the estimates and respective intervals vs. trial number and comment on their stability. Compare the LS estimates and confidence interval, to the overall Boostrap estimates and confidence interval.

The following code segments is used to estimate the Bootstrap estimates and their respective confidence intervals.

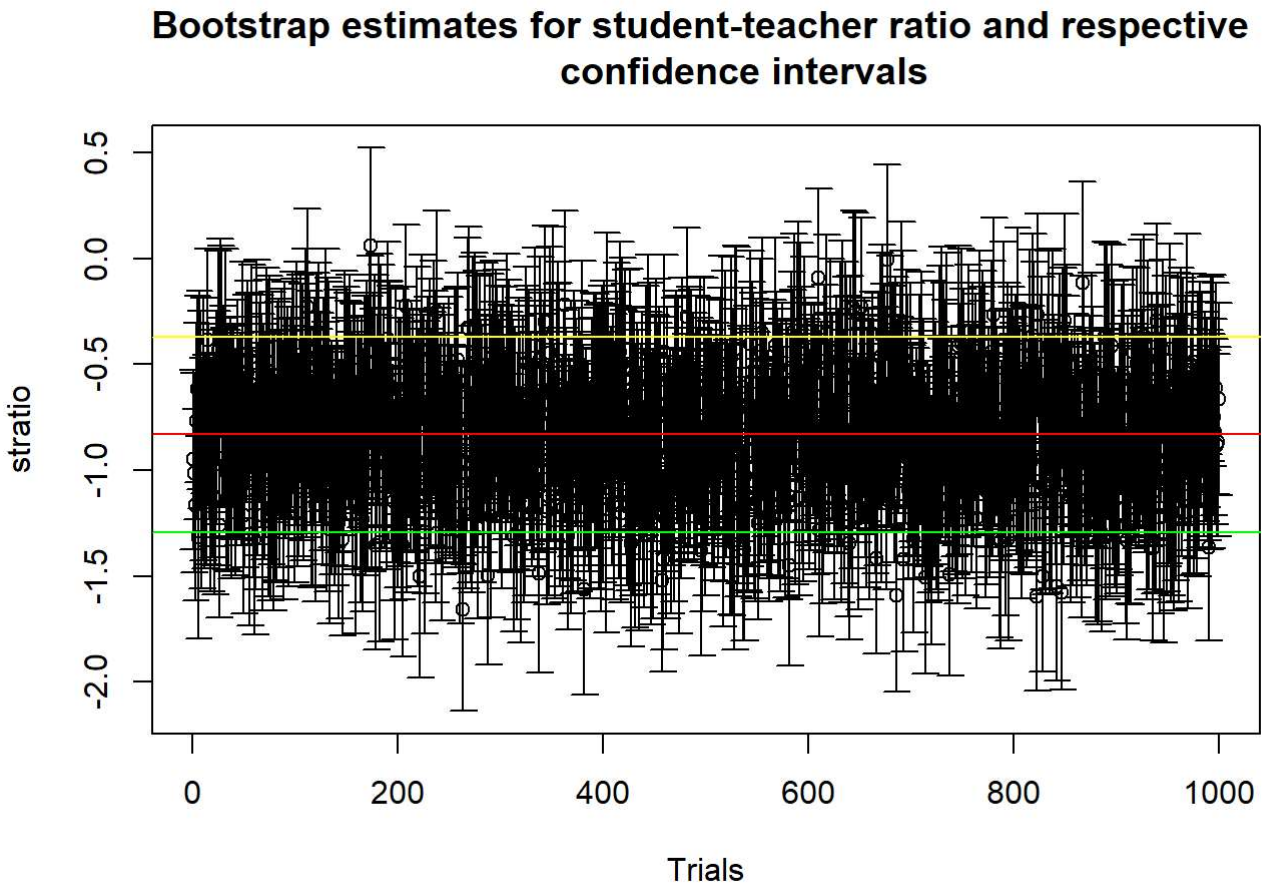
```
set.seed(3244)

#create null lists to store, beta estimates and confidence intervals
bstar=NULL
conf1 = NULL
conf2 = NULL
conf3 = NULL
conf4 = NULL

n=length(CASchools$avgscore)
B=1000                                                    #1000 trials

for(draw in 1:B){
  Dstar = CASchools[sample(1:n,size=n,replace=TRUE),]     #sample from original data
  mod =lm(avgscore~stratio+income+lunch,data=Dstar)       #run linear regression
  bstar = rbind(bstar,coef(mod))                          #store the estimates

  #store confidence intervals for each parameter
  conf1 = rbind(conf1,confint(mod,'(Intercept)',level=0.95))
  conf2 = rbind(conf2,confint(mod,'stratio',level=0.95))
  conf3 = rbind(conf3,confint(mod,'income',level=0.95))
  conf4 = rbind(conf4,confint(mod,'lunch',level=0.95))

}
```

Now we plot the estimates and their respective intervals vs. trial number with the following code:

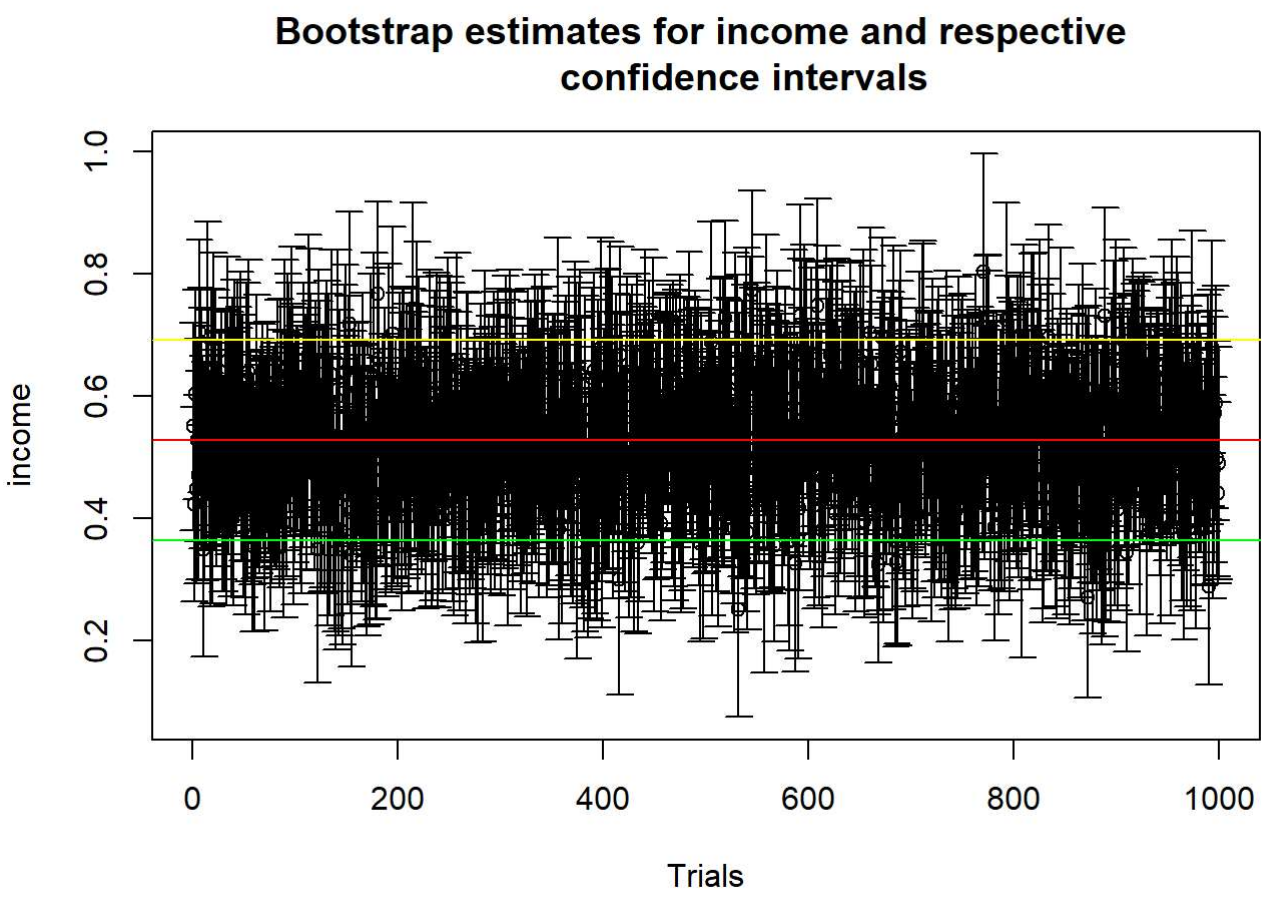
```
plot.new()
#Plot for Intercept estimates
plotCI(1:1000,bstar[,1],ui=conf1[,2],li=conf1[,1],ylab= "Intercept",xlab="Trials",
main="Bootstrap estimates for Intercept and respective
       confidence intervals")
abline(h=685.0311786, col="red")                    #LS estimate
abline(h=695.4681,col="yellow")                     #LS CI upper interval
abline(h=674.5943,col="green")                      #LS CI lower interval
```



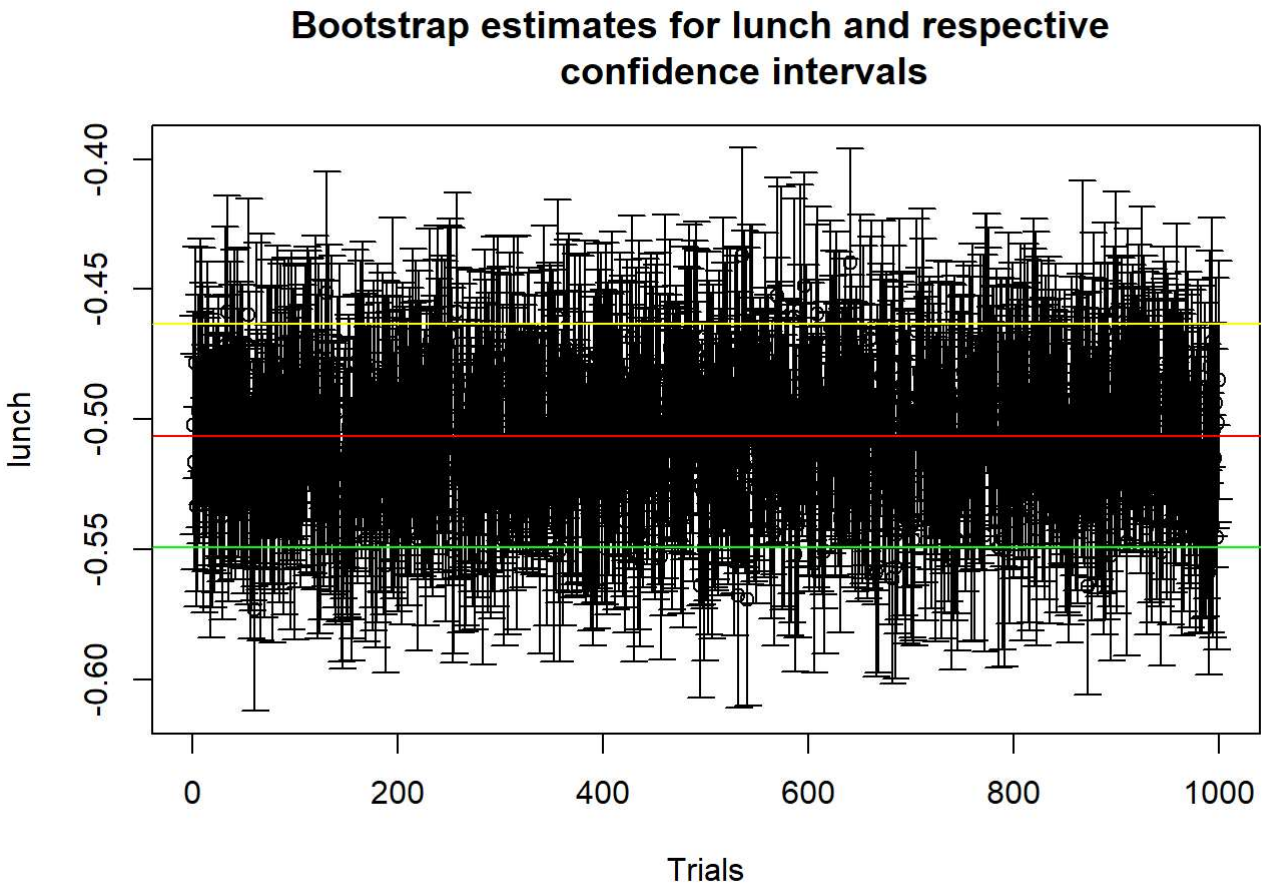

```
plot.new()
#Plot for student-teacher ratio estimates
plotCI(1:1000,bstar[,2],ui=conf2[,2],li=conf2[,1],ylab= "stratio",xlab="Trials",
main="Bootstrap estimates for student-teacher ratio and respective
       confidence intervals")
abline(h=-0.8307316, col="red")
abline(h=-0.3702454,col="yellow")
abline(h=-1.291218,col="green")
```




```
plot.new()
#Plot for income estimates
plotCI(1:1000,bstar[,3],ui=conf3[,2],li=conf3[,1],ylab= "income",xlab="Trials",
main="Bootstrap estimates for income and respective
       confidence intervals")
abline(h= 0.5272695, col="red")
abline(h=0.6911125,col="yellow")
abline(h=0.3634264,col="green")
```




```
plot.new()
#Plot for lunch estimates
plotCI(1:1000,bstar[,4],ui=conf4[,2],li=conf4[,1],ylab= "lunch",xlab="Trials",
main="Bootstrap estimates for lunch and respective
       confidence intervals")
abline(h= -0.5063093, col="red")
abline(h=-0.4634597,col="yellow")
abline(h=-0.549159,col="green")
```



The red,yellow and green lines on the charts are the LS estimates, upper limit of the confidence interval, and lower limit of the confidence interval respectively.

As we can see, the bootstrap estimates are all mostly stable with the majority of the data falling within the same range as the LS estimates.

(b) Using MCMC generate appropriate distributions for the model parameters and use these to perform a Bayesian regression and the respective 95% credible intervals. Plot the respective parameter posterior distributions with their credible intervals overlaid on the histograms. Compare the LS estimates and confidence interval, to the Bayesian results.

In order to perform a Bayesian regression using MCMC we use the MCMCregress function in R. Then we check the estimates and credible intervals.

```
#run Bayesian regressian using MCMC
lm.MCMC <- MCMCregress(avgscore~stratio+income+lunch, data = CASchools, burnin=100,
                       mcmc=1000, thin=1,b0=c(0,0,0,0),B0=c(0,0,0,0),c0=0.001, d0=0.001)

summary(lm.MCMC)                                    #view summary of estimates
```

```
##
## Iterations = 101:1100
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean      SD  Naive SE Time-series SE
## (Intercept) 684.9434 5.12890 0.1621902      0.1521668
## stratio      -0.8255 0.22374 0.0070754      0.0067080
## income        0.5283 0.08717 0.0027566      0.0027566
## lunch        -0.5067 0.02224 0.0007034      0.0007034
## sigma2       77.9920 5.25873 0.1662955      0.1662955
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%      50%      75%    97.5%
## (Intercept) 675.1338 681.6706 685.0561 688.2199 695.1191
## stratio      -1.2864  -0.9638  -0.8322  -0.6770  -0.3674
## income        0.3536   0.4732   0.5302   0.5873   0.6924
## lunch        -0.5503  -0.5213  -0.5059  -0.4911  -0.4634
## sigma2       68.6137  74.4085  77.7721  81.2660  88.5200
```

```
HPDinterval(lm.MCMC)                          #view credible intervals
```

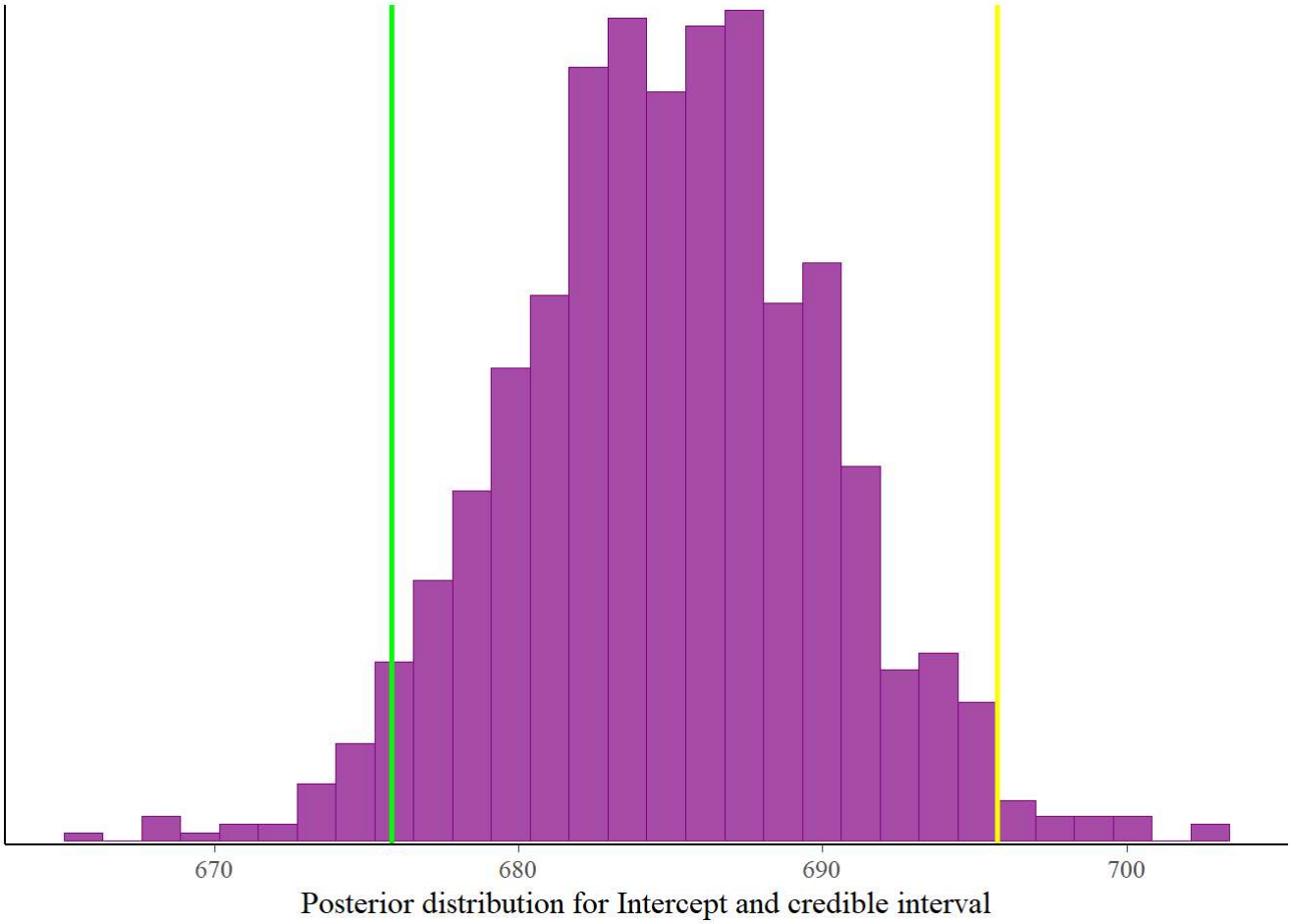
```
##                   lower       upper
## (Intercept) 675.8146633 695.7474219
## stratio      -1.2214597  -0.3171554
## income        0.3637874   0.6963552
## lunch        -0.5471539  -0.4615774
## sigma2       68.5012225  88.1528403
## attr(,"Probability")
## [1] 0.95
```

If we compare the estimates to our LS and Bootstrapping estimates in part (a) we notice that our estimates are very similar.

Now we plot the respective parameter posterior distributions with their credible intervals overlaid on the histograms.

```
posterior<-as.array(lm.MCMC)

color_scheme_set("purple")
p<-mcmc_hist(posterior,pars=c("(Intercept)"))
p + vline_at(675.8146633, linetype = 1, size = 1, color = "green")+ vline_at(695.7474219,
linetype = 1, size = 1, color = "yellow") +
xlab("Posterior distribution for Intercept and credible interval")
```

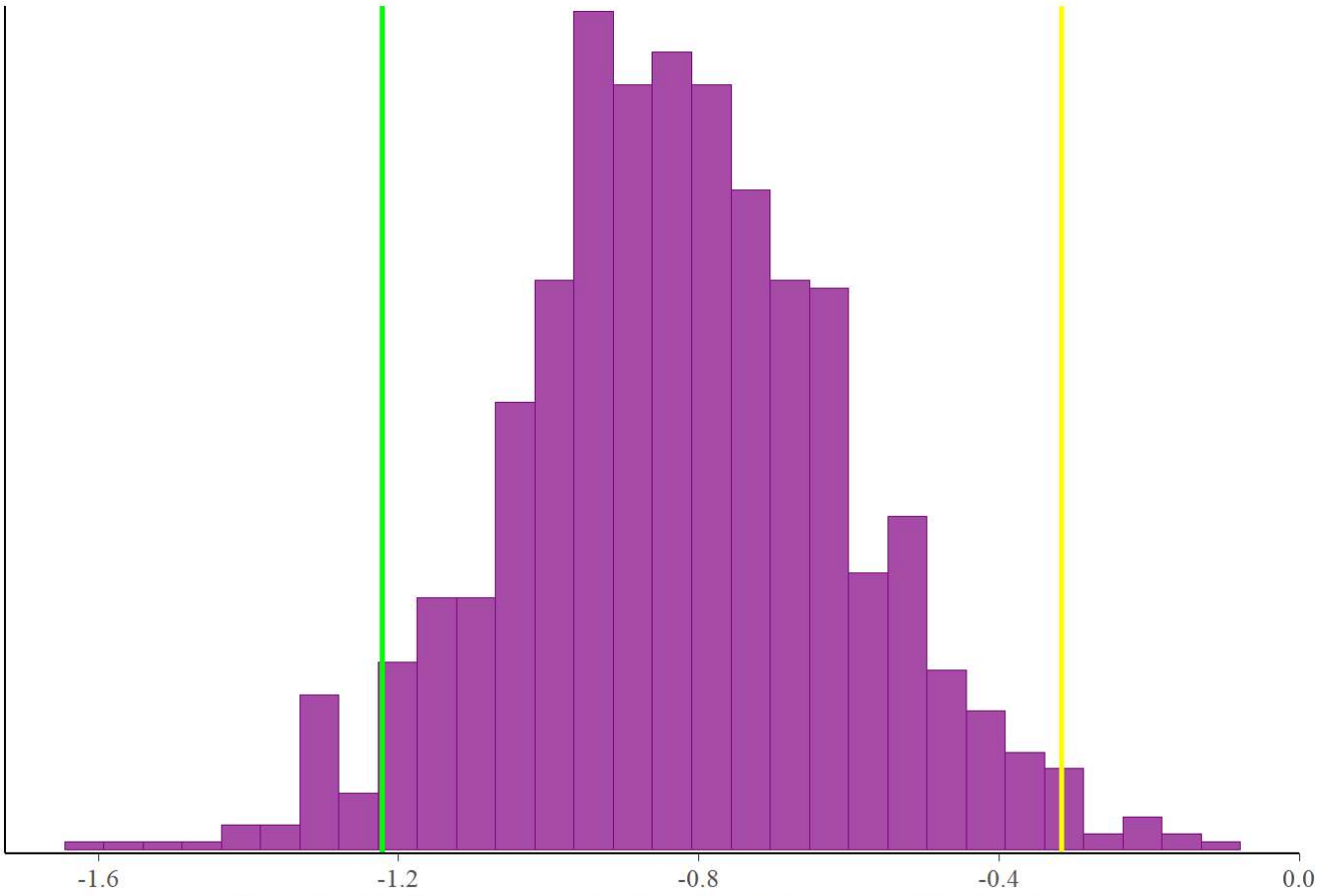
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Posterior distribution for Intercept and credible interval

```
color_scheme_set("purple")
q<-mcmc_hist(posterior,pars=c("stratio"))
q + vline_at(-1.2214597, linetype = 1, size = 1, color = "green") + vline_at(-0.3171554,
                                                 linetype = 1, size = 1, color = "yellow") +
xlab("Posterior distribution for student-teacher ratio and credible interval")
```

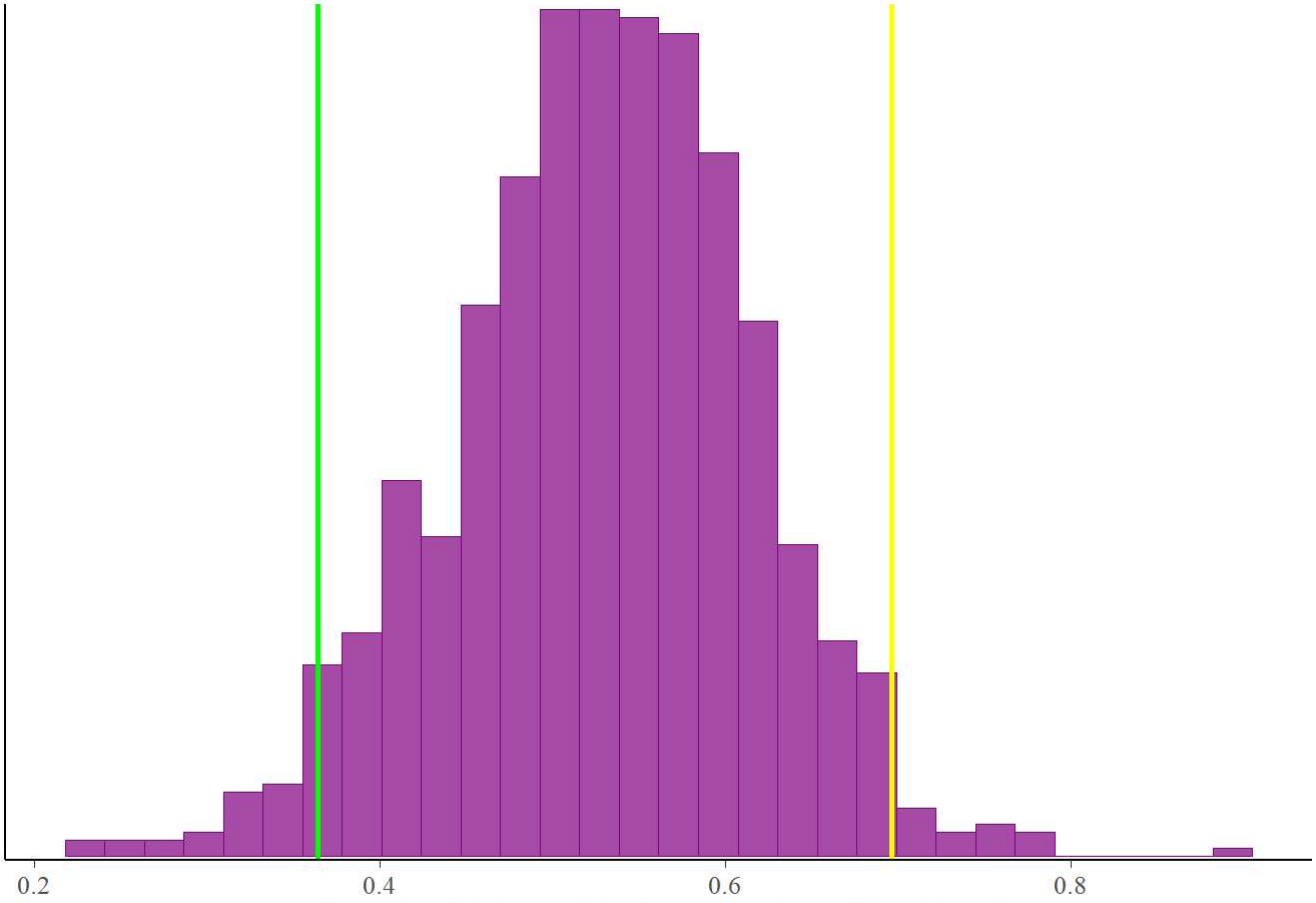
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Posterior distribution for student-teacher ratio and credible interval

```
color_scheme_set("purple")
r<-mcmc_hist(posterior,pars=c("income"))
r + vline_at(0.3637874, linetype = 1, size = 1, color = "green") + vline_at(0.6963552,
linetype = 1, size = 1, color = "yellow") +
xlab("Posterior distribution for income and credible interval")
```

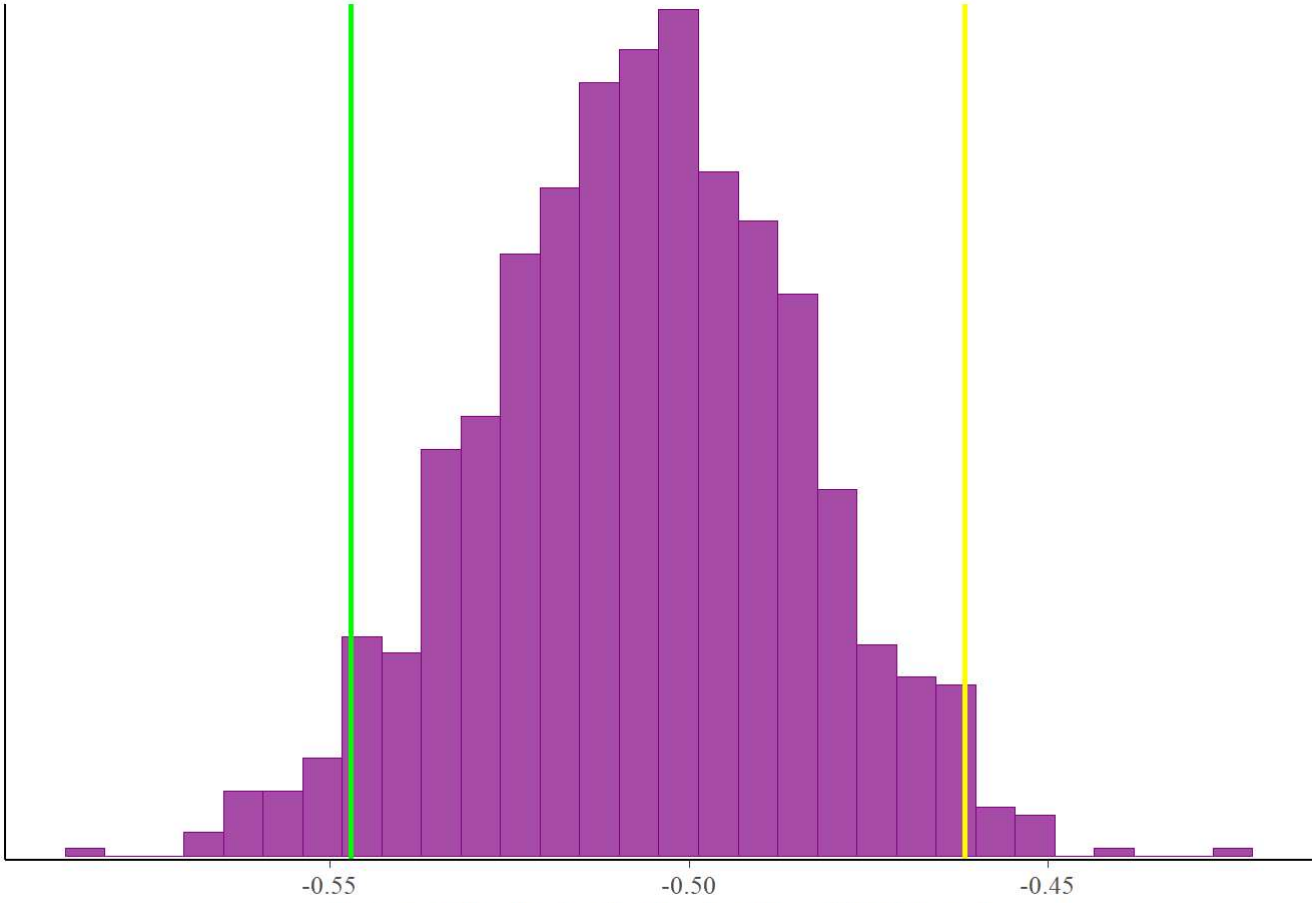
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Posterior distribution for income and credible interval

```
color_scheme_set("purple")
s<-mcmc_hist(posterior,pars=c("lunch"))
s + vline_at(-0.5471539, linetype = 1, size = 1, color = "green") + vline_at(-0.4615774,
linetype = 1, size = 1, color = "yellow")+
xlab("Posterior distribution for lunch and credible interval")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Posterior distribution for lunch and credible interval

As previously mentioned these estimates are very similar to those of our LS estimates.

(c) Based on your results from the 3 methods, provide a convincing argument for which method you believe is better, and why.

While the estimates produced from LS and from the Bayesian regression are very similar, we believe that the Bayesian method is better. The LS assumes that there is enough data to make meaningful interpretations while Bayes updates this belief with the prior and likelihood function. We believe that since it is accounting for more information and continuously updating itself to account for new information, then it is more accurate.

**Question 2: Fittind Distributions

First we will create a vector that contains the recession dates.

```
recession.dates.1 <- data.frame(dates = seq(as.Date("1953-07-01"),
                           as.Date("1954-05-01"), by = "month"))
recession.dates.2 <- data.frame(dates = seq(as.Date("1957-08-01"),
                           as.Date("1958-04-01"), by = "month"))
recession.dates.3 <- data.frame(dates = seq(as.Date("1960-04-01"),
                           as.Date("1961-02-01"), by = "month"))
recession.dates.4 <- data.frame(dates = seq(as.Date("1969-12-01"),
                           as.Date("1970-11-01"), by = "month"))
recession.dates.5 <- data.frame(dates = seq(as.Date("1973-11-01"),
                           as.Date("1975-03-01"), by = "month"))
recession.dates.6 <- data.frame(dates = seq(as.Date("1980-01-01"),
                           as.Date("1980-07-01"), by = "month"))
recession.dates.7 <- data.frame(dates = seq(as.Date("1981-07-01"),
                           as.Date("1982-11-01"), by = "month"))
recession.dates.8 <- data.frame(dates = seq(as.Date("1990-07-01"),
                           as.Date("1991-03-01"), by = "month"))
recession.dates.9 <- data.frame(dates = seq(as.Date("2001-03-01"),
                           as.Date("2001-11-01"), by = "month"))
recession.dates.10 <- data.frame(dates = seq(as.Date("2007-12-01"),
                           as.Date("2009-06-01"), by = "month"))


recession.dates <- rbind(recession.dates.1, recession.dates.2, recession.dates.3,
                         recession.dates.4, recession.dates.5, recession.dates.6,
                         recession.dates.7, recession.dates.8, recession.dates.9,
                         recession.dates.10)
```

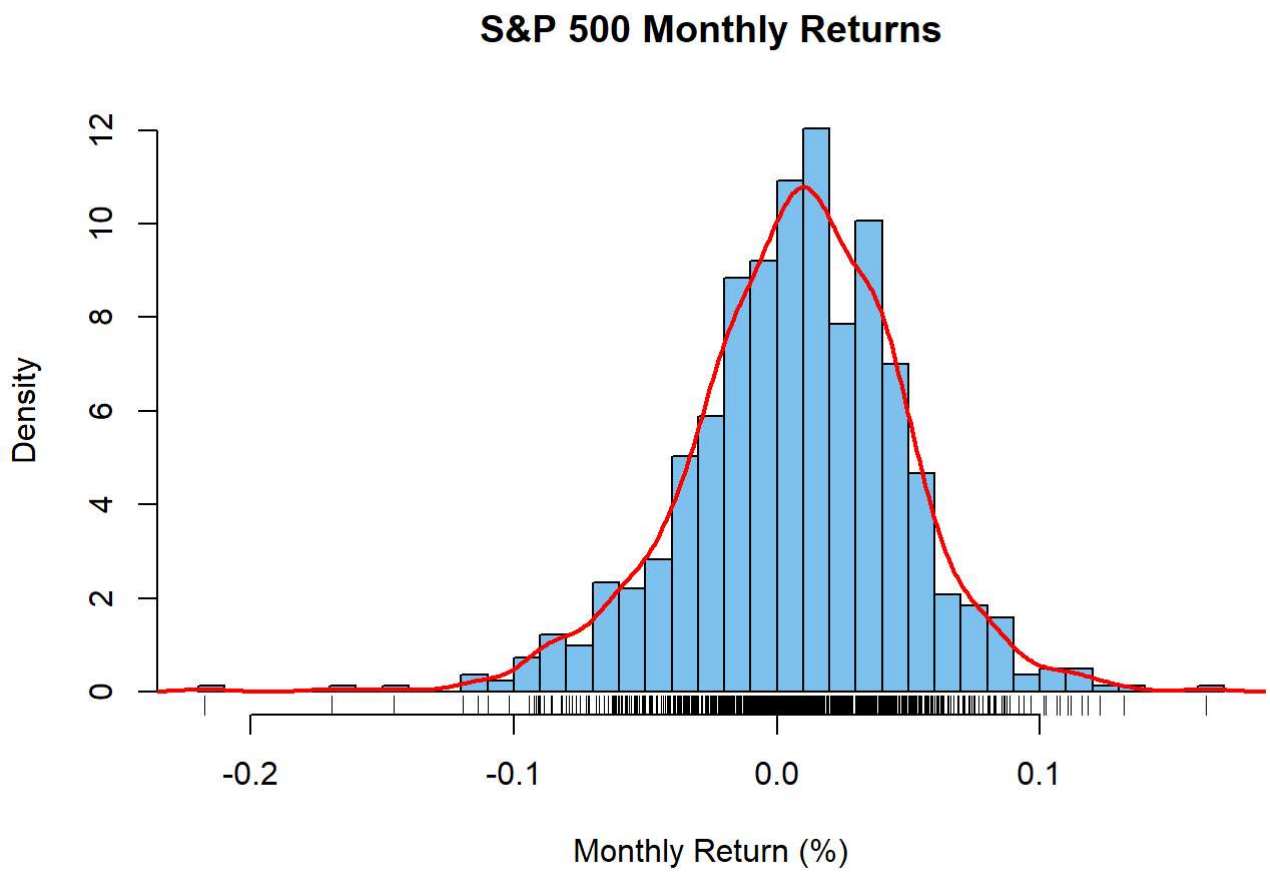
Problem 2a: S&P 500 Monthly Returns

Plot the histogram of the data and overlay the respective density curve.

```
sp <- read.csv("GSPC Yahoo.csv", header = TRUE)
sp <- sp[, c(1,6)]
sp$ret <- Delt(sp$Adj.Close)
sp <- sp[-1, 3]
sp <- as.vector(sp)
length(sp)
```

```
## [1] 815
```

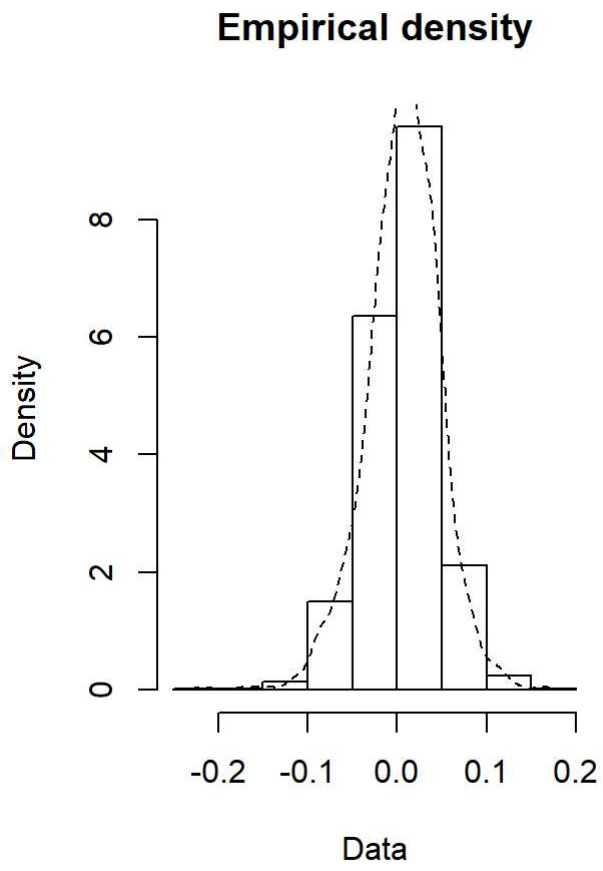
```
par(mfrow = c(1,1))
hist(sp, breaks = "FD", prob = TRUE, main = "S&P 500 Monthly Returns", col = "skyblue2",
     xlab = "Monthly Return (%)")
lines(density(sp), col = "red", lwd = 2)
rug(sp)
```

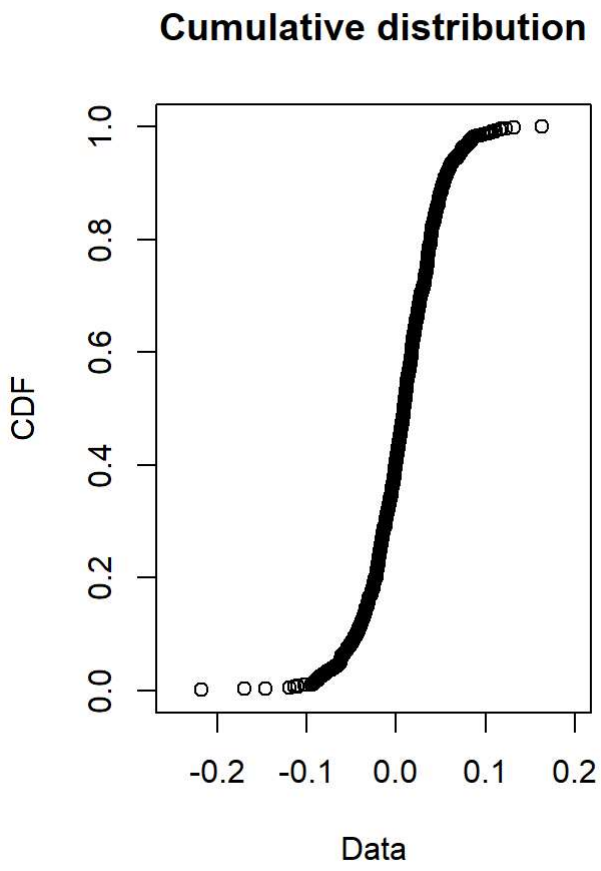


Problem 2b: S&P 500 Monthly Returns

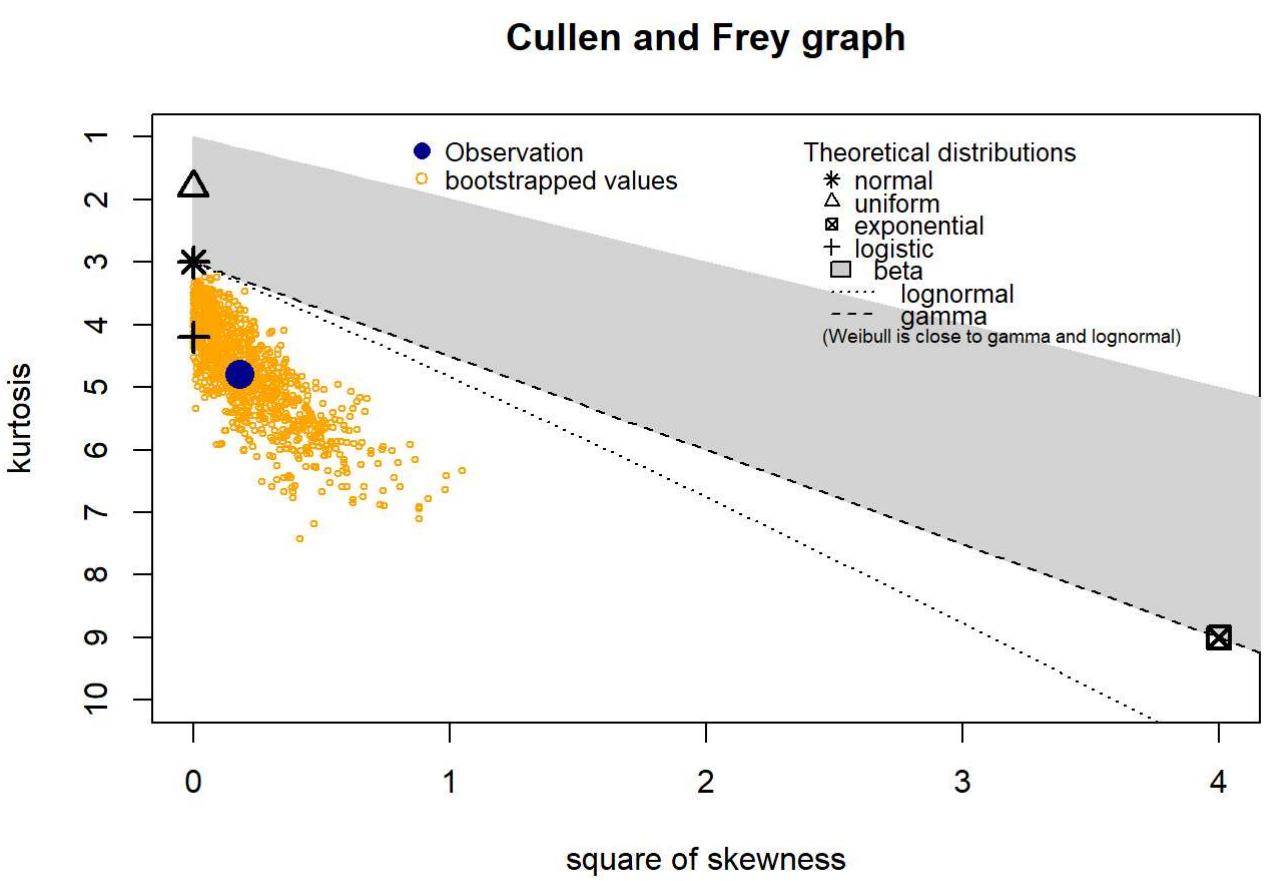
Fit a distribution to the histogram found in part (a).

```
plotdist(sp, histo = TRUE, demp = TRUE)
```






```
descdist(sp, boot = 1000)
```




```
## summary statistics
## ------
## min:  -0.2176304   max:  0.1630469
## median:  0.009105594
## mean:  0.006900958
## estimated sd:  0.04117175
## estimated skewness:  -0.4256281
## estimated kurtosis:  4.789791
```

From the above test, we determine that the data comes from a normal or a logistic distribution.

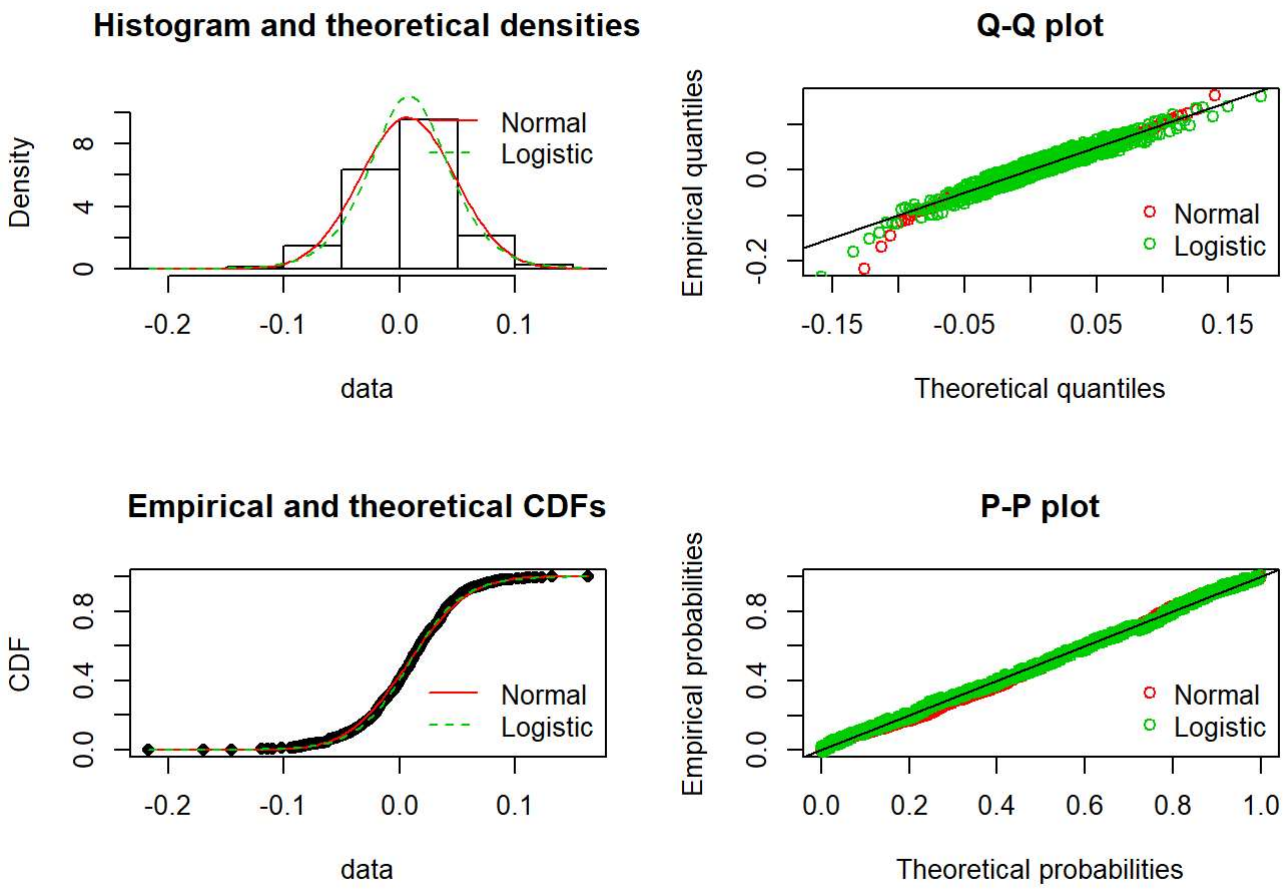
```
sp.fitreturn.norm <- fitdist(as.numeric(sp), "norm")
sp.fitreturn.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##         estimate  Std. Error
## mean 0.006900958 0.001441298
## sd   0.041146481 0.001016444
```

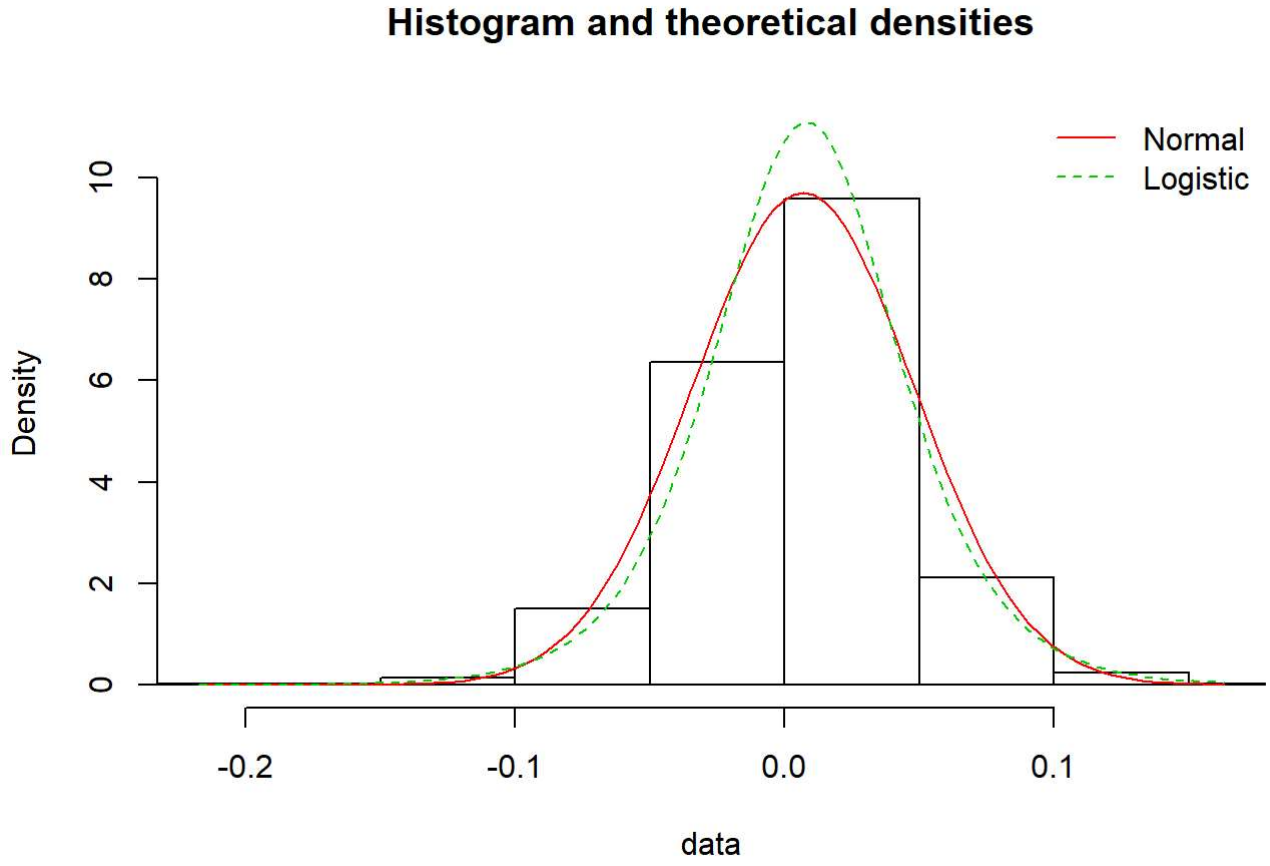
```
sp.fitreturn.logis <- fitdist(as.numeric(sp), "logis")
sp.fitreturn.logis
```

```
## Fitting of the distribution ' logis ' by maximum likelihood
## Parameters:
##             estimate   Std. Error
## location 0.008069425 0.0013663890
## scale    0.022529324 0.0006559134
```

```
par(mfrow=c(2,2))
plot.legend <- c("Normal", "Logistic")
denscomp(list(sp.fitreturn.norm, sp.fitreturn.logis), legendtext = plot.legend)
qqcomp(list(sp.fitreturn.norm, sp.fitreturn.logis), legendtext = plot.legend)
cdfcomp(list(sp.fitreturn.norm, sp.fitreturn.logis), legendtext = plot.legend)
ppcomp(list(sp.fitreturn.norm, sp.fitreturn.logis), legendtext = plot.legend)
```




```
par(mfrow=c(1,1))
denscomp(list(sp.fitreturn.norm, sp.fitreturn.logis), legendtext = plot.legend)
```



It appears that the fitted logistic distribution more precisely describes the tails of the data. Finally, we perform a KS Test on the data with the normal distribution and the logistic distribution, using the parameters found through Maximum Likelihood Estimation. Using a KS Test, the null hypothesis is that the data comes from the same distribution as the fitted distribution. The alternative hypothesis therefore is that the data does not come from the same distribution as the fitted distribution.

```
ks.test(sp, "pnorm", mean = sp.fitreturn.norm[["estimate"]][["mean"]],
        sd = sp.fitreturn.norm[["estimate"]][["sd"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sp
## D = 0.041861, p-value = 0.1149
## alternative hypothesis: two-sided
```

```
ks.test(sp, "plogis", location = sp.fitreturn.logis[["estimate"]][["location"]],
        scale = sp.fitreturn.logis[["estimate"]][["scale"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sp
## D = 0.020973, p-value = 0.866
## alternative hypothesis: two-sided
```

For both tests the p-value is high, indicating that we fail to reject the null in both cases. However, the p-value is significantly higher when tested against the logistic distribution. Therefore, we determine that the data follows a logistic distribution. Based on the Maximum Likelihood Estimates for the parameters, the data follows a logistic distribution with location = 0.0081 and scale = 0.0225

Problem 2c: S&P 500 Monthly Returns

Subset the data to include only data points generated during recession periods. Plot a histogram of the data and overlay the respective density curve. Then fit a distribution to the histogram.

```
sp <- read.csv("GSPC Yahoo.csv", header = TRUE)
sp <- sp[, c(1,6)]
date <- as.Date(sp$Date, format = "%Y-%m-%d")
sp <- cbind(date, sp)
sp <- sp[, -2]
sp$ret <- Delt(sp$Adj.Close)
sp <- sp[-1, c(1,3)]
names(sp) <- paste(c("Date", "Monthly Return"))
head(sp)
```

```
nrow(sp)
```

```
## [1] 815
```

```
sp2 <- sp[sp$Date %in% recession.dates$dates, ]
head(sp2)
```

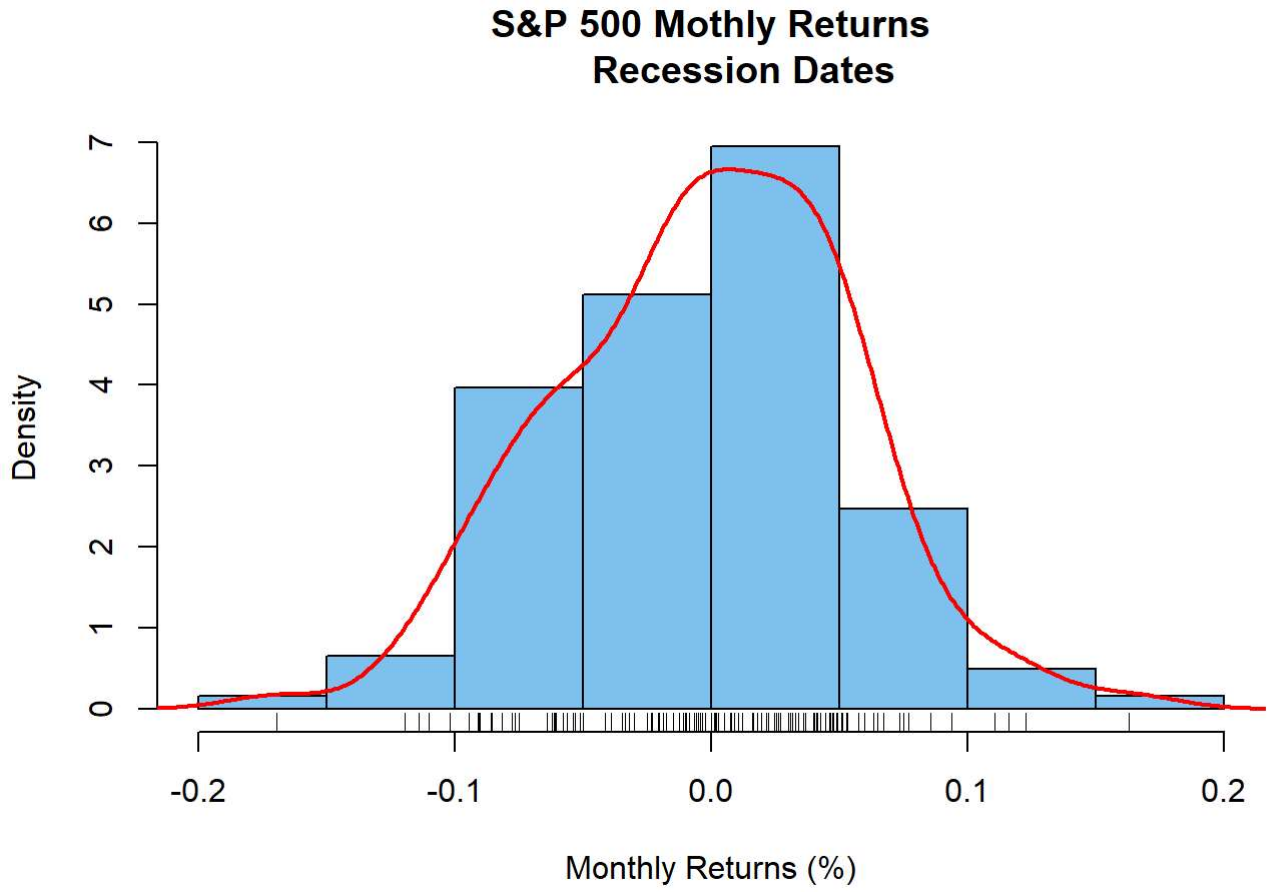
```
nrow(sp2)
```

```
## [1] 121
```

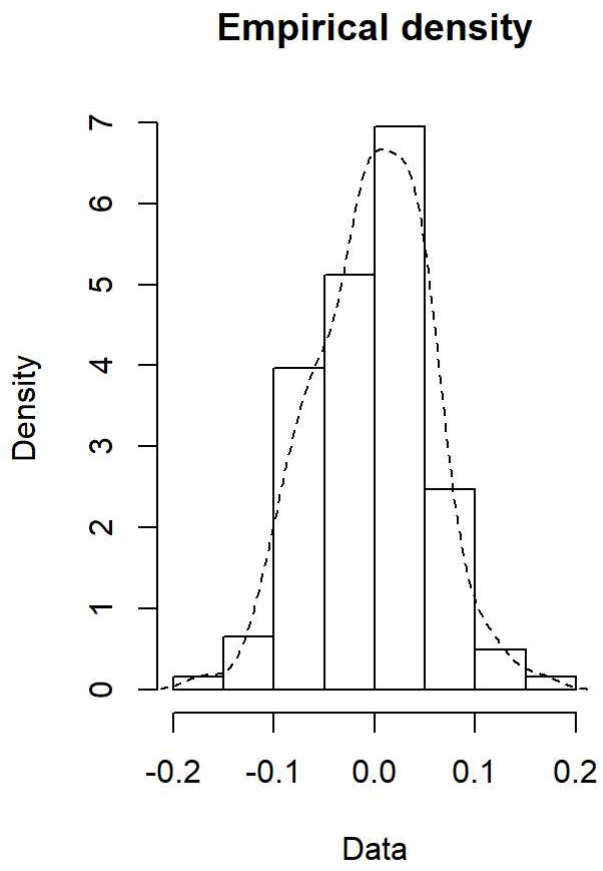
```
sp2 <- sp2[,2]
sp2 <- as.vector(sp2)
head(sp2)
```

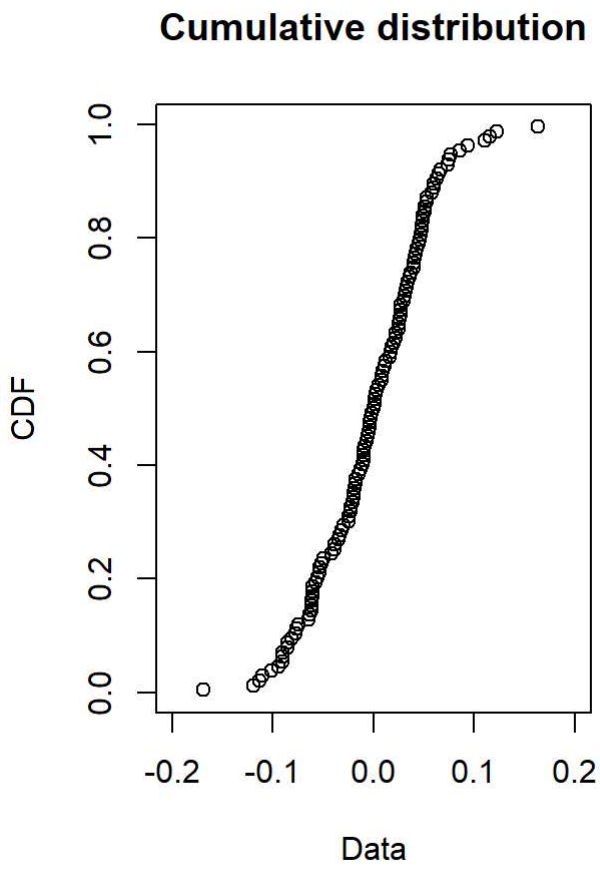
```
## [1]  0.025269305 -0.057777778  0.001286449  0.050963640  0.008964914
## [6]  0.002019346
```

```
par(mfrow = c(1,1))
hist(sp2, breaks = "FD", prob = TRUE, col = "skyblue2",  main = "S&P 500 Mothly Returns
      Recession Dates", xlab = "Monthly Returns (%)")
lines(density(sp2), col = "red", lwd = 2)
rug(sp2)
```

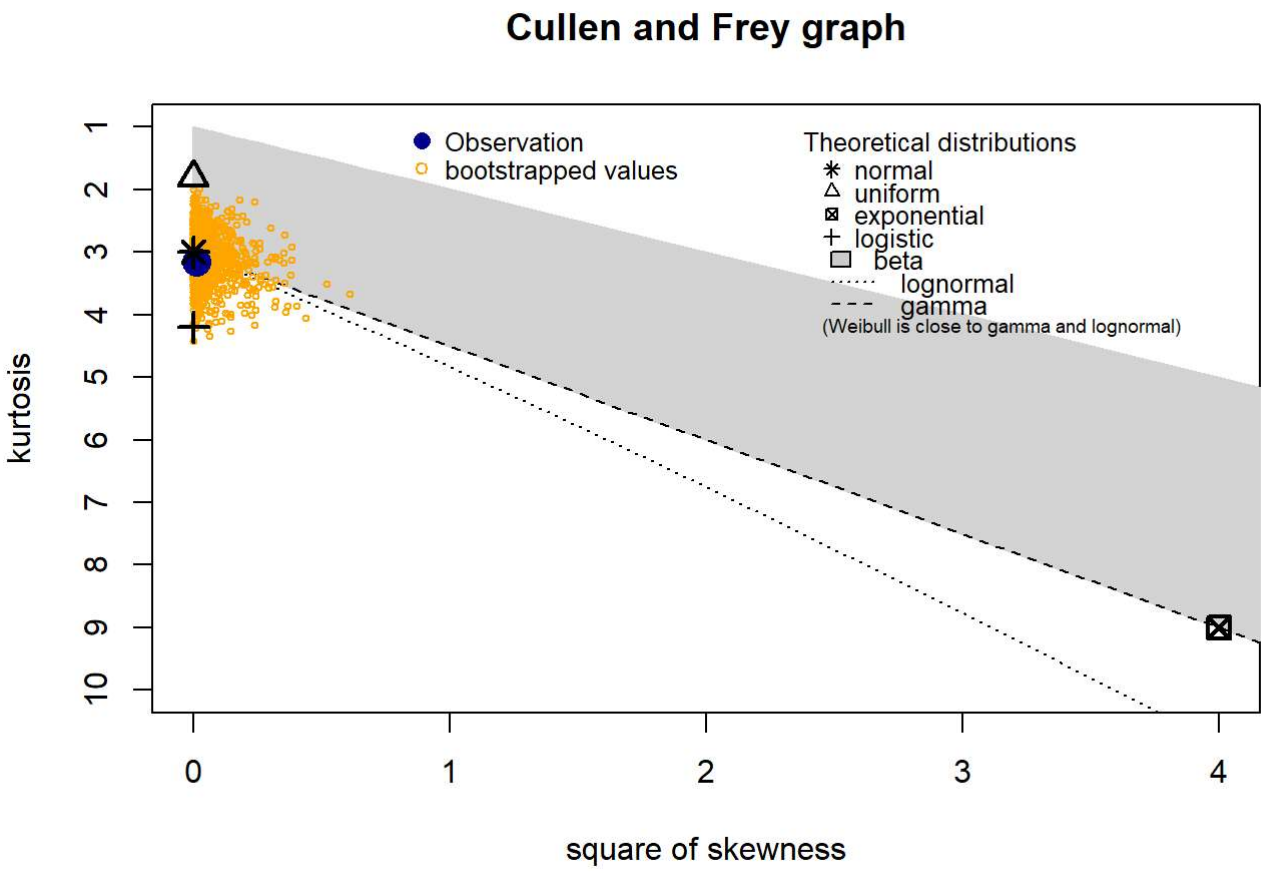



```
plotdist(sp2, histo = TRUE, demp = TRUE)
```






```
descdist(sp2, boot = 1000)
```

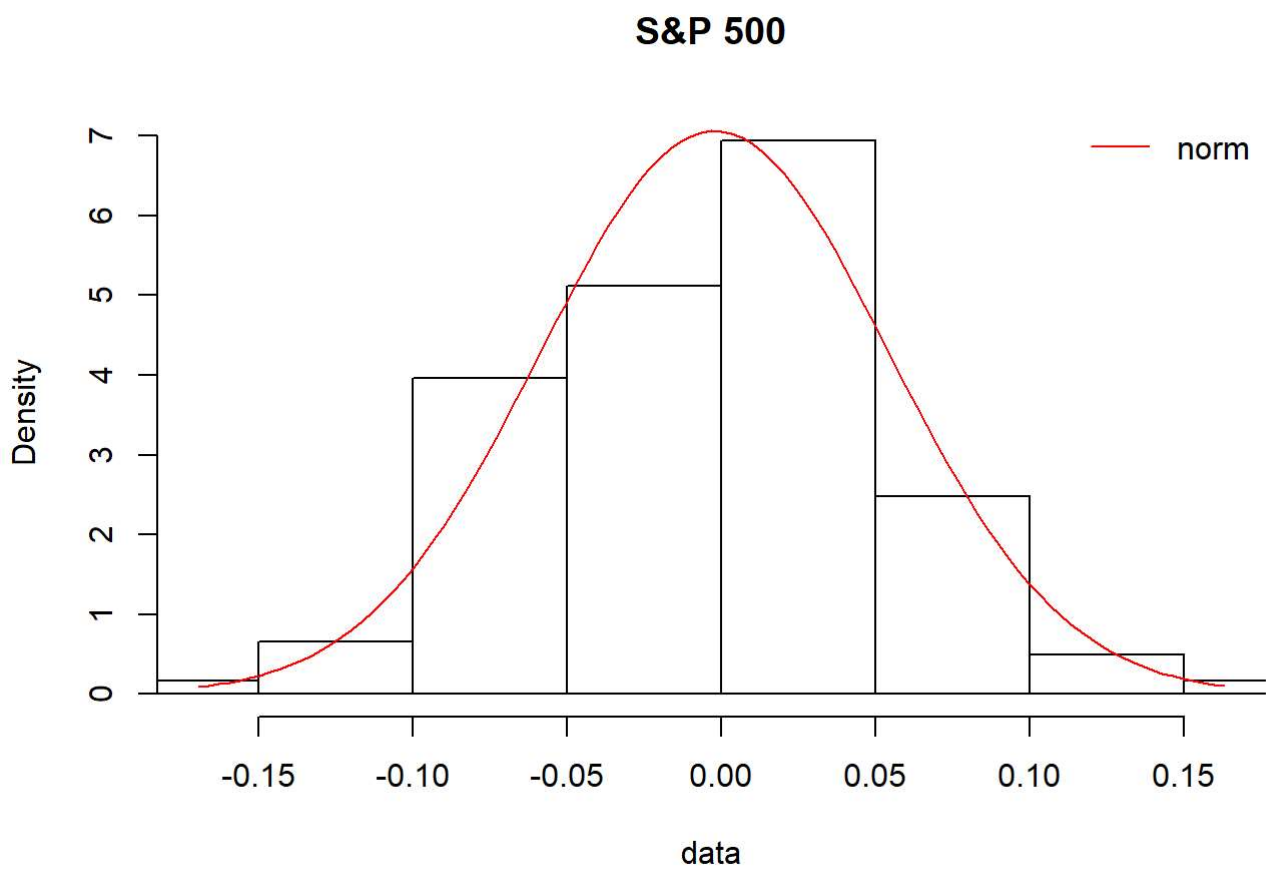



```
## summary statistics
## ------
## min:  -0.1694245   max:  0.1630469
## median:  0.0001958265
## mean:  -0.002073451
## estimated sd:  0.05670787
## estimated skewness:  -0.1009994
## estimated kurtosis:  3.152819
```

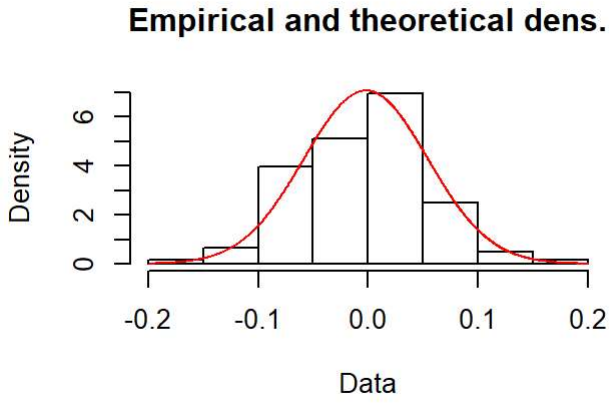
```
sp2.fitreturn <- fitdist(as.numeric(sp2), "norm")
sp2.fitreturn
```

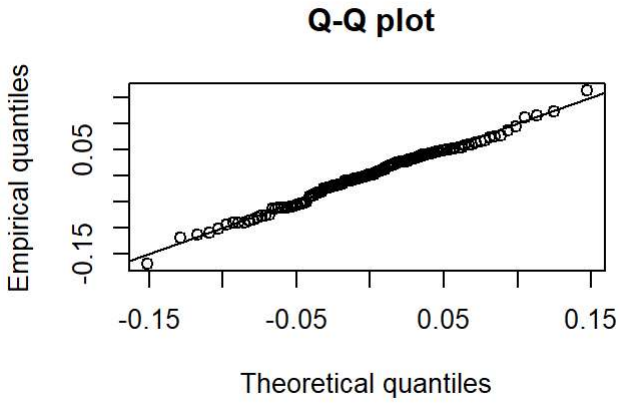
```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##          estimate  Std. Error
## mean -0.002073451 0.005133914
## sd    0.056473058 0.003625105
```

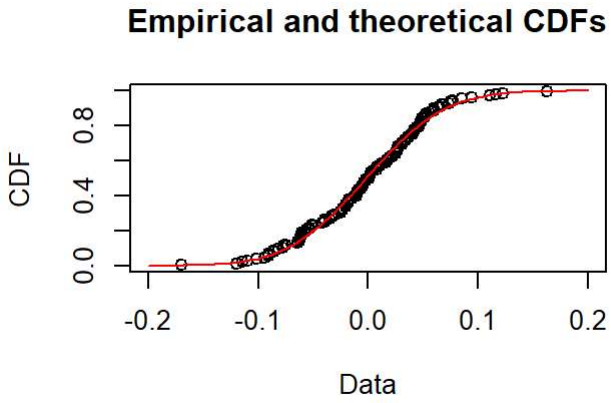
```
denscomp(sp2.fitreturn, main = "S&P 500")
```

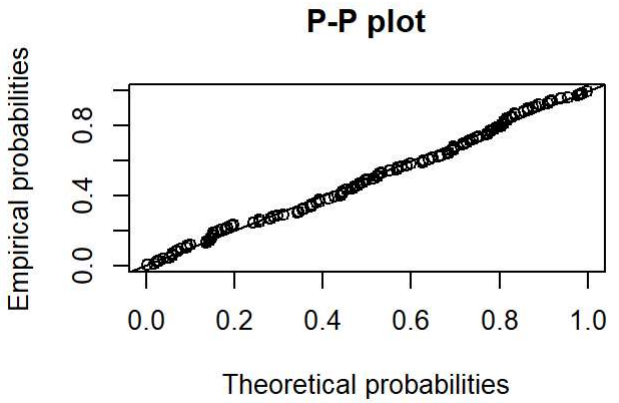



```
plot(sp2.fitreturn)
```










```
ks.test(sp2, "pnorm", mean = sp2.fitreturn[["estimate"]][["mean"]],
        sd = sp2.fitreturn[["estimate"]][["sd"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sp2
## D = 0.046726, p-value = 0.9543
## alternative hypothesis: two-sided
```

We determine using the Cullen and Frey Graph that the data comes from a normal distribuiton and test this using a KS Test against a normal distribution with the parameters estimated using Maximum Likelihood Estimation. The p-value is very high, indicating that we fail to reject the null and the data comes from the fitted normal distribution. Therefore, we determine that the data containing only points generated during recession periods follows a normal distribution. Based on the Maximum Liklehood Estimates of the parameters, the data follows a normal distribution with mean -0.0021 and sd = 0.0564

Problem 2d: S&P 500 Monthly Returns

Repeat par (c) with the data subsetted to include only data points generated during non-recession periods.

```
sp <- read.csv("GSPC Yahoo.csv", header = TRUE)
sp <- sp[, c(1,6)]
date <- as.Date(sp$Date, format = "%Y-%m-%d")
sp <- cbind(date, sp)
sp <- sp[, -2]
sp$ret <- Delt(sp$Adj.Close)
sp <- sp[-1, c(1,3)]
names(sp) <- paste(c("Date", "Monthly Return"))
head(sp)
```

```
nrow(sp)
```

```
## [1] 815
```

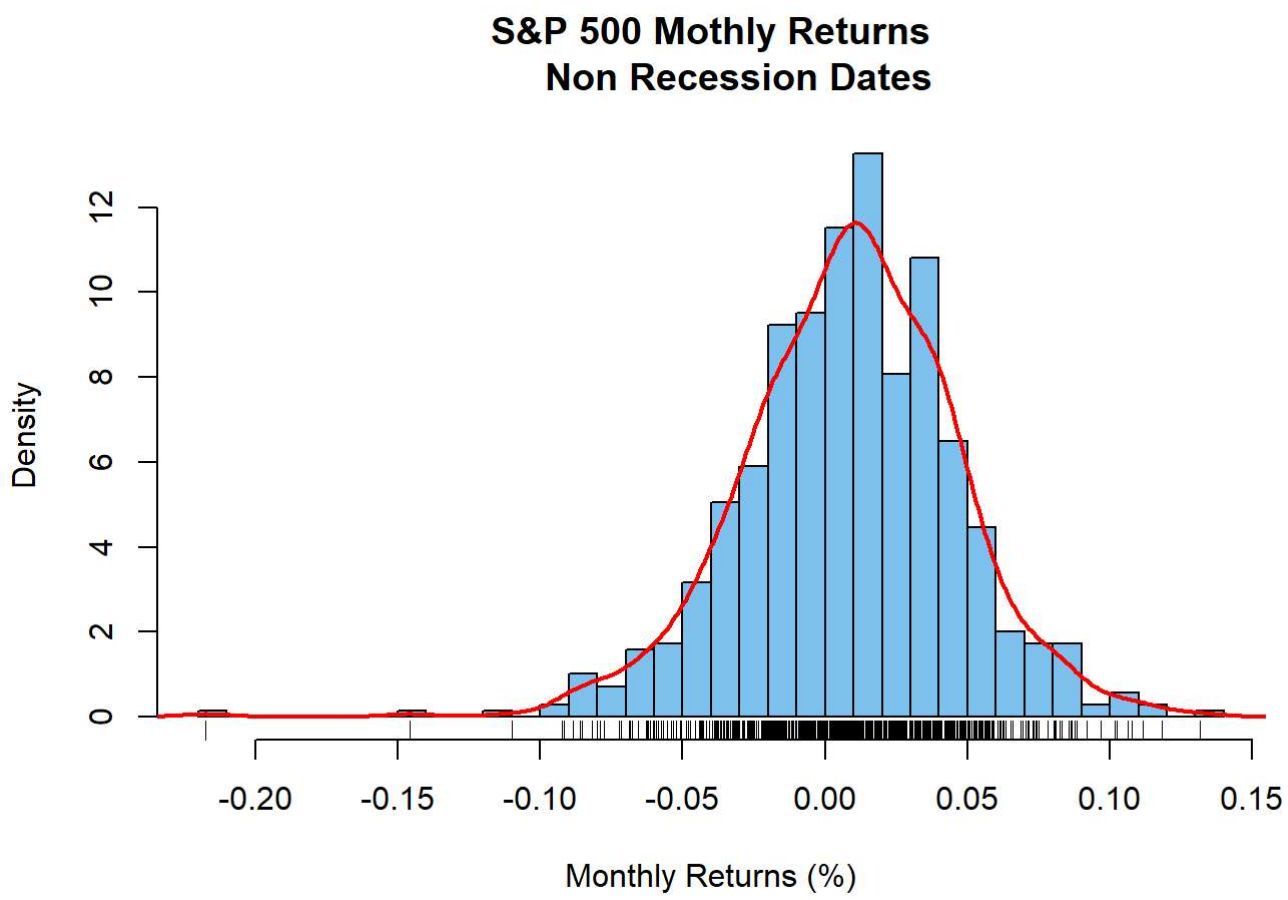
```
sp3 <- sp[!(sp$Date %in% recession.dates$dates), ]
head(sp3)
```

```
nrow(sp3)
```

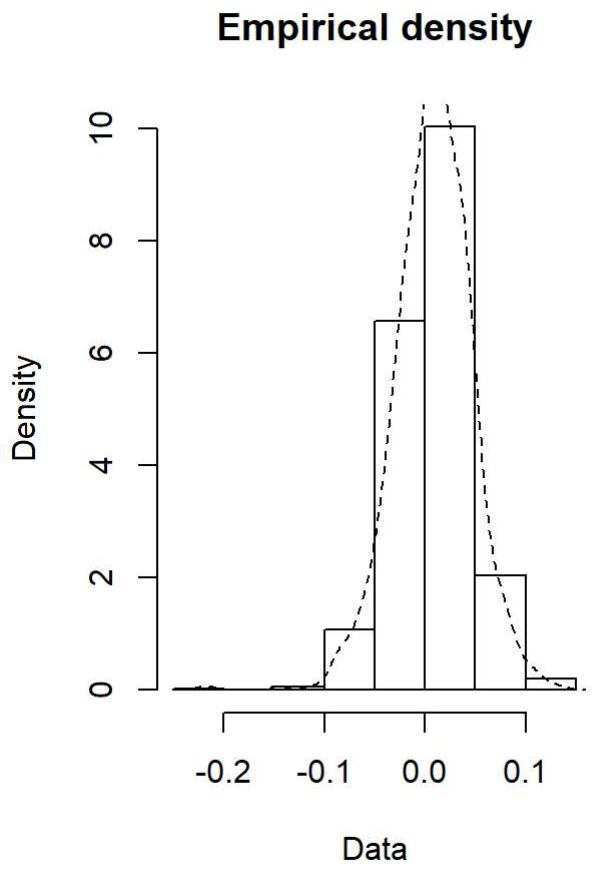
```
## [1] 694
```

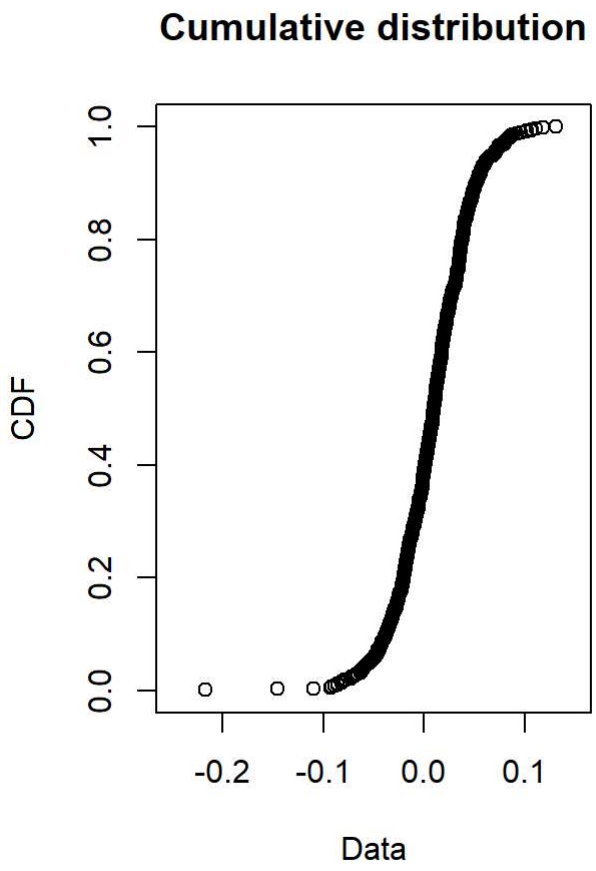
```
sp3 <- sp3[,2]
sp3 <- as.vector(sp3)
```

```
hist(sp3, breaks = "FD", prob = TRUE, col = "skyblue2",  main = "S&P 500 Mothly Returns
     Non Recession Dates", xlab = "Monthly Returns (%)")
lines(density(sp3), col = "red", lwd = 2)
rug(sp3)
```

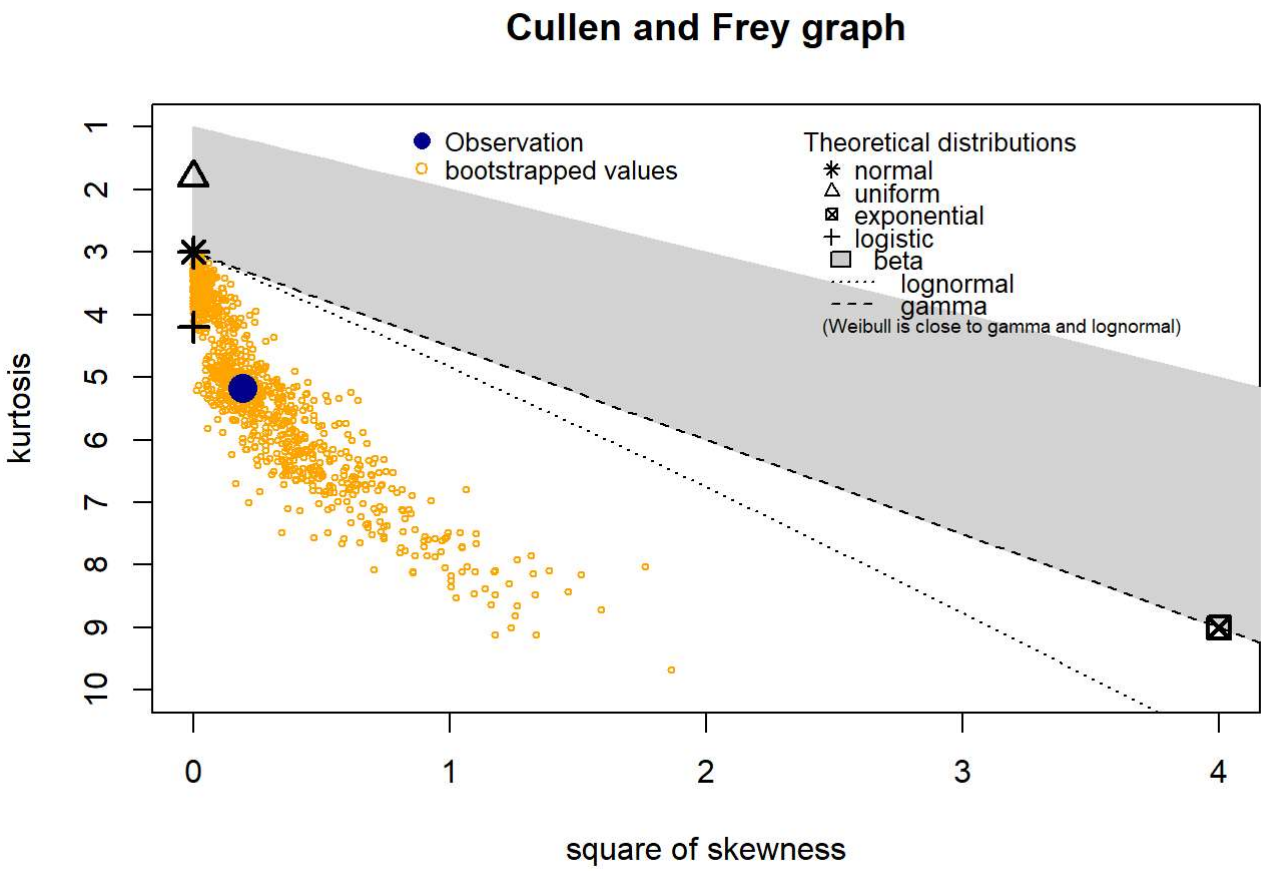



```
plotdist(sp3, histo = TRUE, demp = TRUE)
```






```
descdist(sp3, boot = 1000)
```




```
## summary statistics
## ------
## min:  -0.2176304   max:  0.1317669
## median:  0.009922507
## mean:  0.00846566
## estimated sd:  0.03765269
## estimated skewness:  -0.4360499
## estimated kurtosis:  5.168837
```

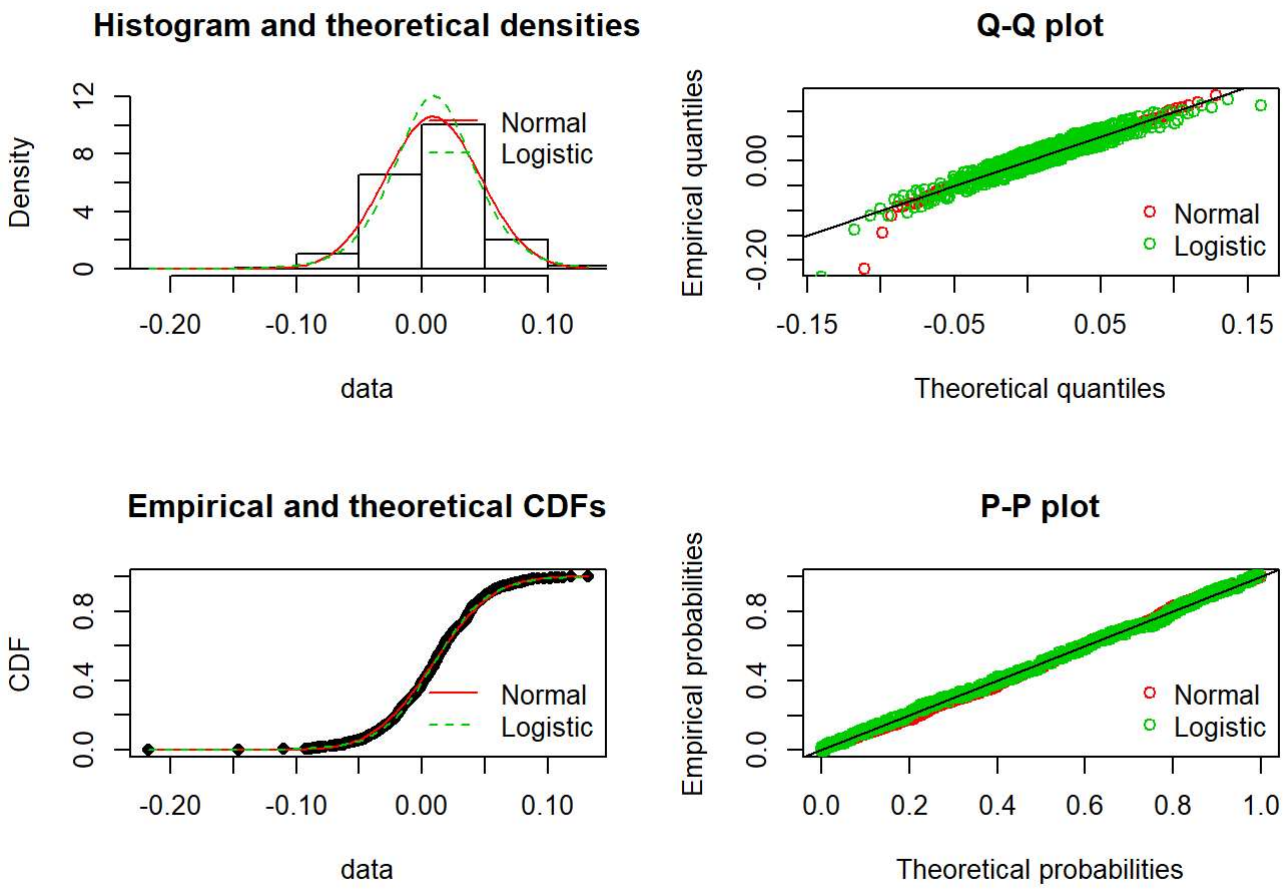
```
sp3.fitreturn.norm <- fitdist(as.numeric(sp3), "norm")
sp3.fitreturn.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##        estimate  Std. Error
## mean 0.00846566 0.001428246
## sd   0.03762555 0.001006714
```

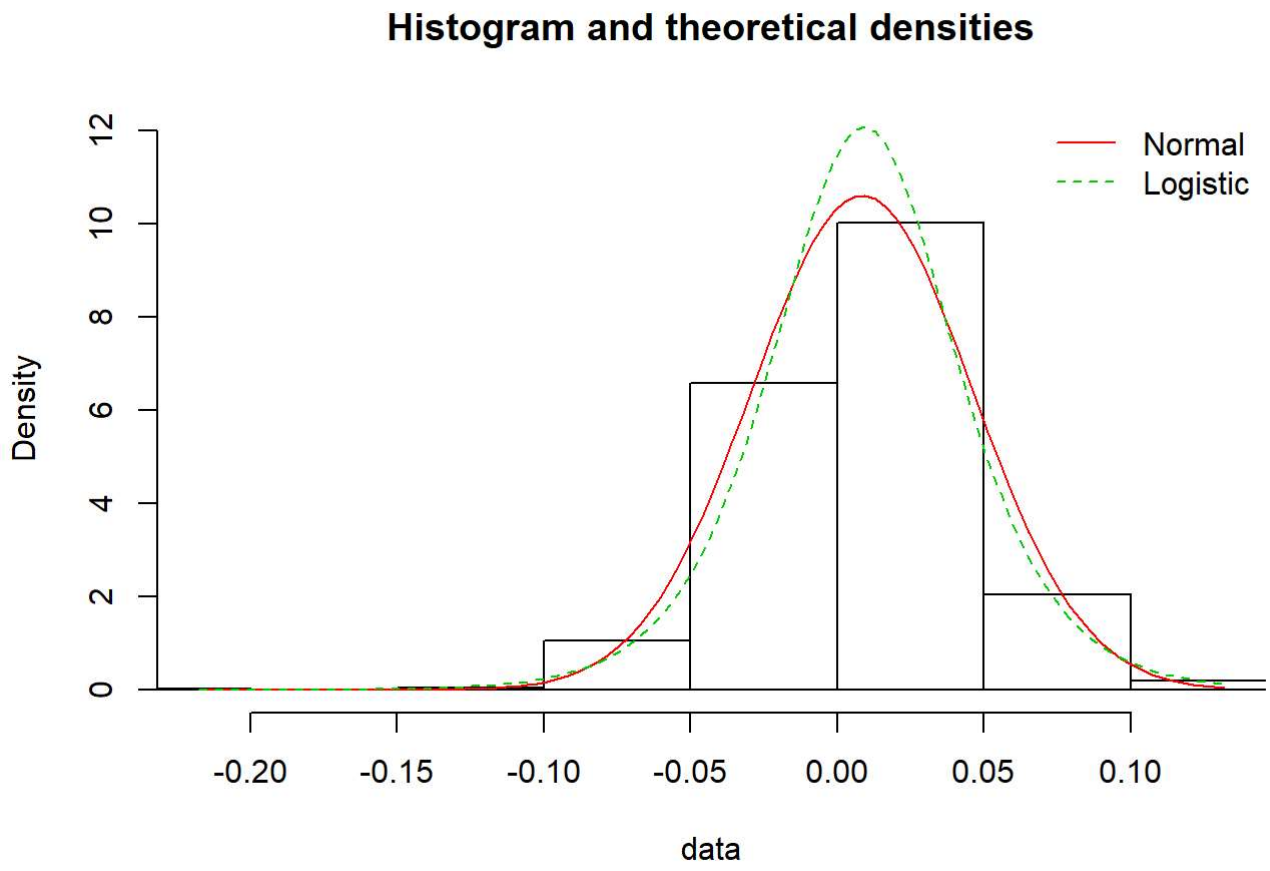
```
sp3.fitreturn.logis <- fitdist(as.numeric(sp3), "logis")
sp3.fitreturn.logis
```

```
## Fitting of the distribution ' logis ' by maximum likelihood
## Parameters:
##             estimate   Std. Error
## location 0.009210443 0.0013636873
## scale    0.020706289 0.0006509965
```

```
par(mfrow=c(2,2))
plot.legend <- c("Normal", "Logistic")
denscomp(list(sp3.fitreturn.norm, sp3.fitreturn.logis), legendtext = plot.legend)
qqcomp(list(sp3.fitreturn.norm, sp3.fitreturn.logis), legendtext = plot.legend)
cdfcomp(list(sp3.fitreturn.norm, sp3.fitreturn.logis), legendtext = plot.legend)
ppcomp(list(sp3.fitreturn.norm, sp3.fitreturn.logis), legendtext = plot.legend)
```




```
par(mfrow=c(1,1))
denscomp(list(sp3.fitreturn.norm, sp3.fitreturn.logis), legendtext = plot.legend)
```



It appears that the fitted logistic distribution more precisely describes the tails of the data. Finally, we perform a KS Test on the data with the normal distribution and the logistic distribution, using the parameters found through Maximum Likelihood Estimation. Using a KS Test, the null hypothesis is that the data comes from the same distribution as the fitted distribution. The alternative hypothesis therefore is that the data does not come from the same distribution as the fitted distribution.

```
ks.test(sp3, "pnorm", mean = sp3.fitreturn.norm[["estimate"]][["mean"]],
        sd = sp3.fitreturn.norm[["estimate"]][["sd"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sp3
## D = 0.036854, p-value = 0.3026
## alternative hypothesis: two-sided
```

```
ks.test(sp3, "plogis", location = sp3.fitreturn.logis[["estimate"]][["location"]],
        scale = sp3.fitreturn.logis[["estimate"]][["scale"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sp3
## D = 0.025271, p-value = 0.7673
## alternative hypothesis: two-sided
```

For both tests the p-value is high, indicating that we fail to reject the null in both cases. However, the p-value is significantly higher when tested against the logistic distribution. Therefore, we determine that the data follows a logistic distribution. Based on the Maximum Likelihood Estimates for the parameters, the data follows a logistic distribution with location = 0.0092 and scale = 0.0207

Problem 2e: S&P 500 Monthly Returns

Based on the findings in parts (a) - (d), are there any noticeable differences between the estimated distributions based on how the data was subsetted.

Based on my findings, when the data contains all data points generated over the time period, the data follows a logistic distribution. When the data contains only points generated during recession periods, the data follows a normal distribution. When the data contains only points generated during non-recession periods, the data follows a logistic distribution.

Problem 2f: S&P 500 Monthly Returns

Confirm your answer in part (e) using the KS Test.

```
ks.test(sp, sp2)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sp and sp2
## D = 0.5, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(sp, sp3)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sp and sp3
## D = 0.49856, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(sp2, sp3)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sp2 and sp3
## D = 0.18203, p-value = 0.002165
## alternative hypothesis: two-sided
```

I perform two-sample KS Tests to determine if the three samples come from the same distributions. The tests indicate that the samples all come from different distributions with different parameters.

Problem 2a: Yield Spread

```
tr.10 <- read.csv("DGS10 Fred.csv", header = TRUE)
tr.10 <- tr.10[-nrow(tr.10), ]
nrow(tr.10)
```

```
## [1] 681
```

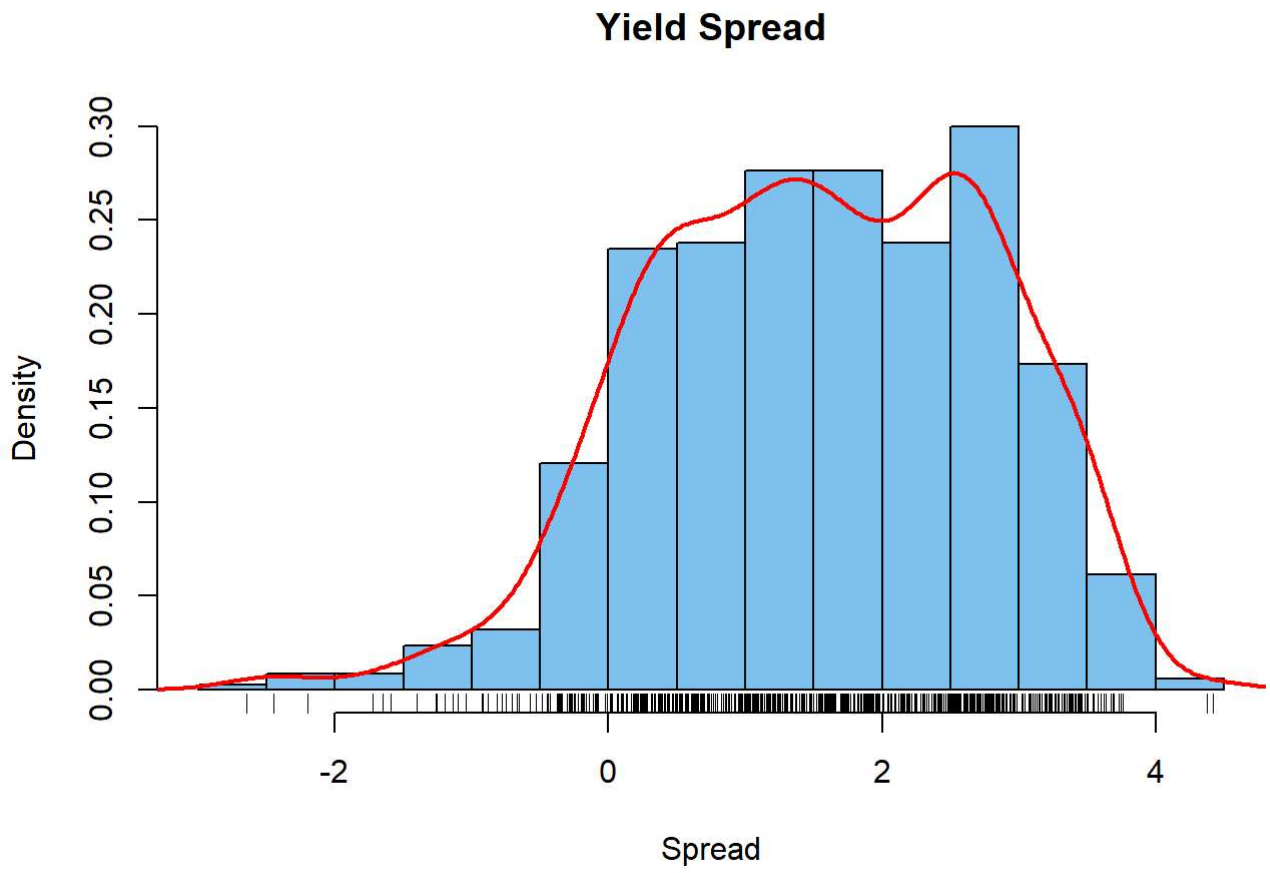
```
tr.3 <- read.csv("3 Month Fred.csv", header = TRUE)
nrow(tr.3)
```

```
## [1] 681
```

```
yield.spread <- cbind(tr.10, tr.3$TB3MS)
date <- as.Date(yield.spread$DATE, format = "%Y-%m-%d")
yield.spread <- cbind(date, yield.spread)
yield.spread <- yield.spread[, -2]
names(yield.spread) <- paste(c("Date", "ten.year", "three.month"))
head(yield.spread)
```

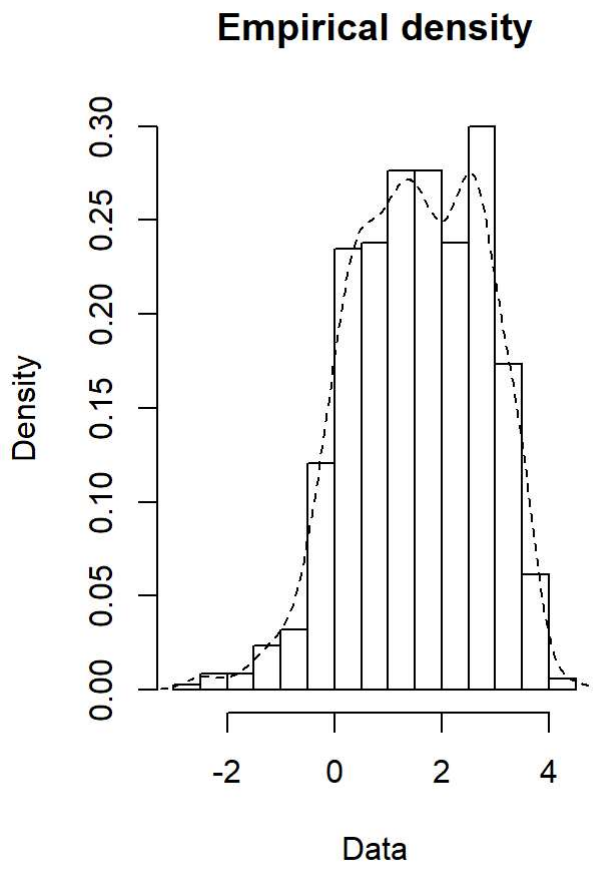
```
yield.spread <- mutate(yield.spread, YS = ten.year - three.month)
yield.spread <- yield.spread[, c(1,4)]
head(yield.spread)
```

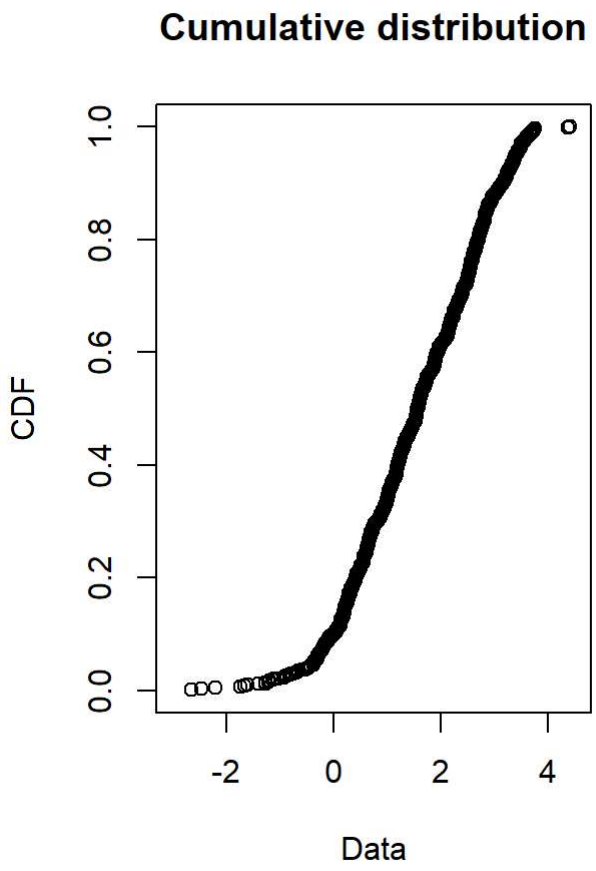
```
ys <- yield.spread[, 2]
ys <- as.vector(ys)
par(mfrow = c(1,1))
hist(ys, breaks = "FD", prob = TRUE, main = "Yield Spread", col = "skyblue2",
     xlab = "Spread")
lines(density(ys), col = "red", lwd = 2)
rug(ys)
```



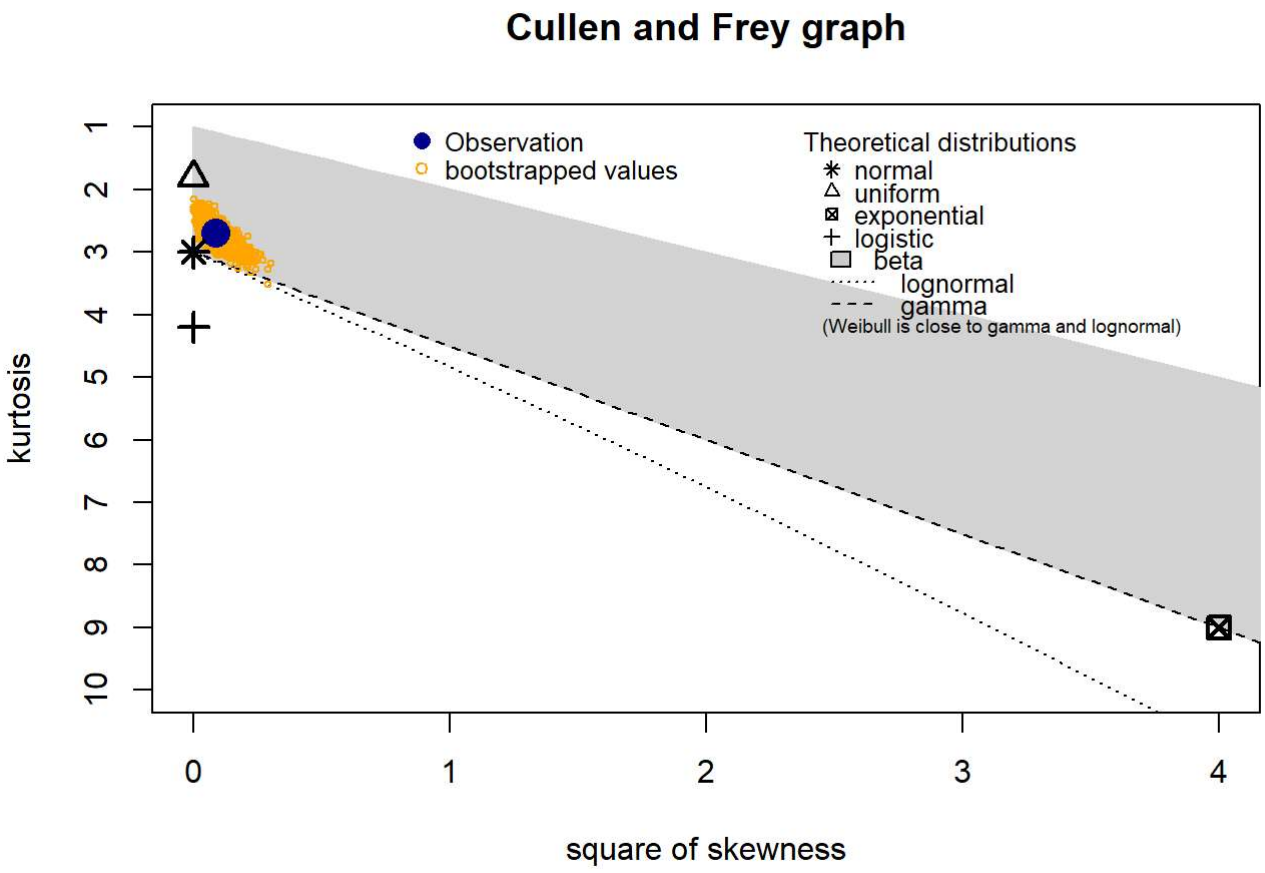
Problem 2b: Yield Spread

```
plotdist(ys, histo = TRUE, demp = TRUE)
```






```
descdist(ys, boot = 1000)
```

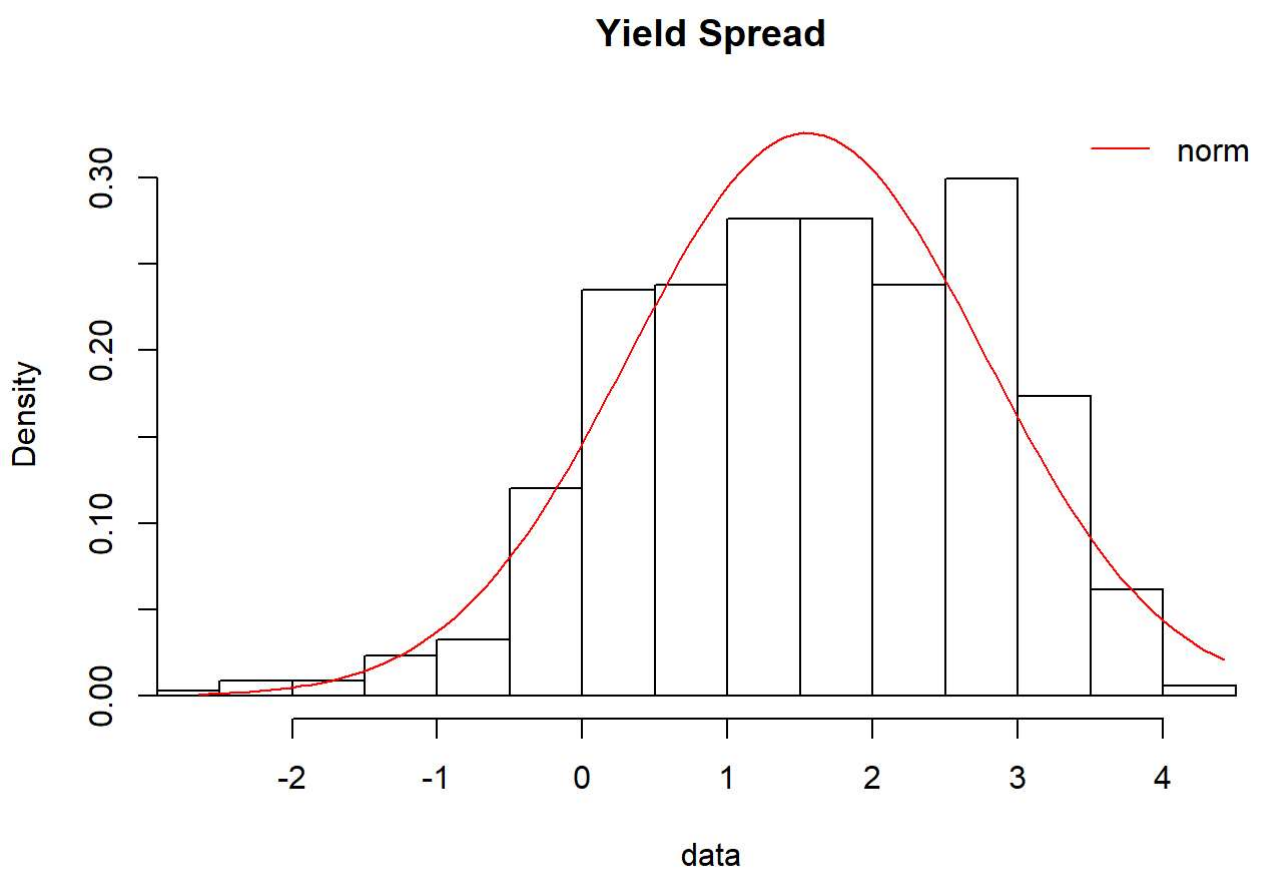



```
## summary statistics
## ------
## min:  -2.645909   max:  4.419048
## median:  1.575238
## mean:  1.547497
## estimated sd:  1.22487
## estimated skewness:  -0.2962264
## estimated kurtosis:  2.69911
```

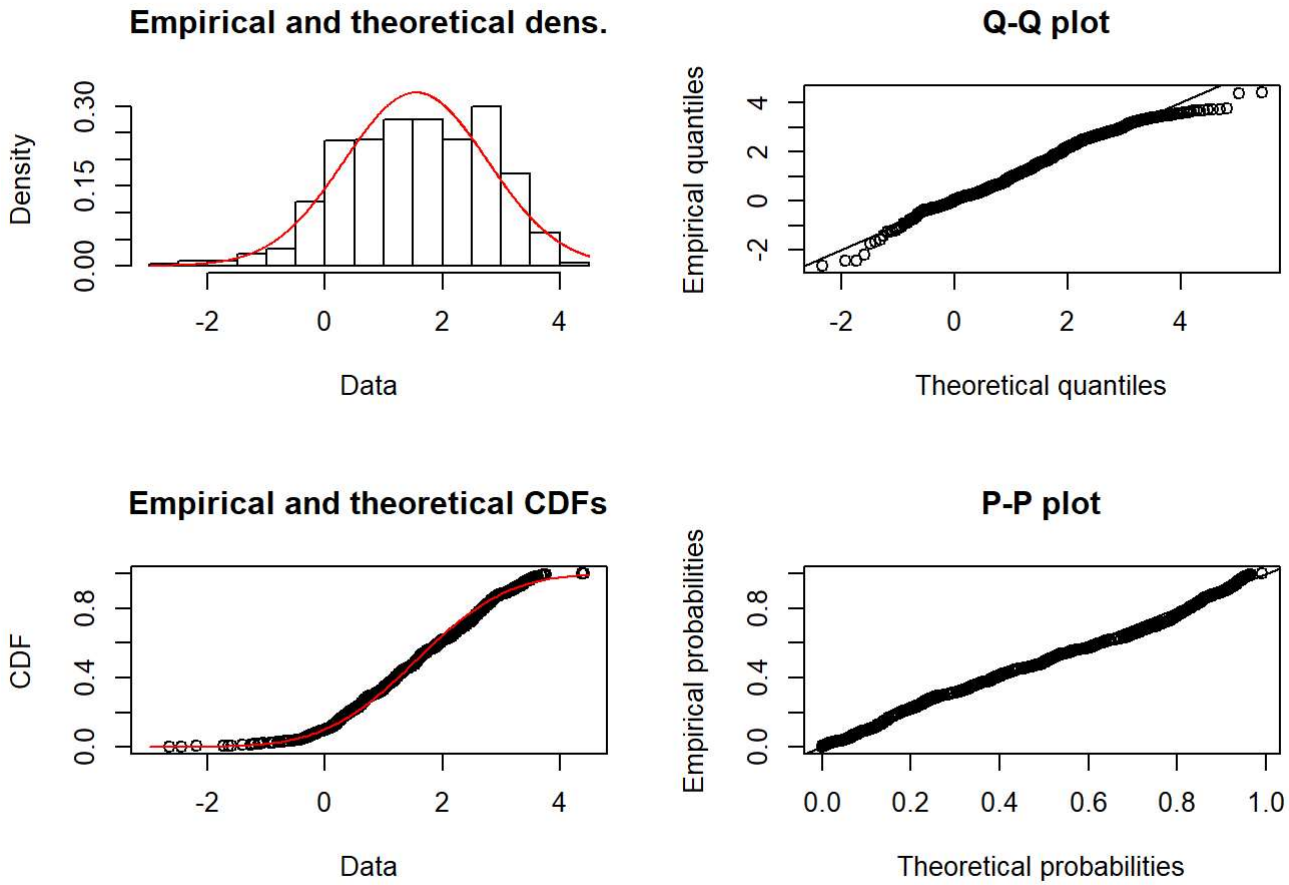
```
ys.fitreturn <- fitdist(as.numeric(ys), "norm")
ys.fitreturn
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 1.547497 0.04690263
## sd   1.223970 0.03316507
```

```
denscomp(ys.fitreturn, main = "Yield Spread")
```




```
plot(ys.fitreturn)
```




```
ks.test(ys, "pnorm", mean = ys.fitreturn[["estimate"]][["mean"]],
        sd = ys.fitreturn[["estimate"]][["sd"]])
```

```
## Warning in ks.test(ys, "pnorm", mean = ys.fitreturn[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  ys
## D = 0.056645, p-value = 0.0253
## alternative hypothesis: two-sided
```

We determine that the data follows a normal distribution. Based on the Maximum Likelihood Estimation for the parameters the data follows a normal distribution with mean = 1.5474 and sd = 1.2239

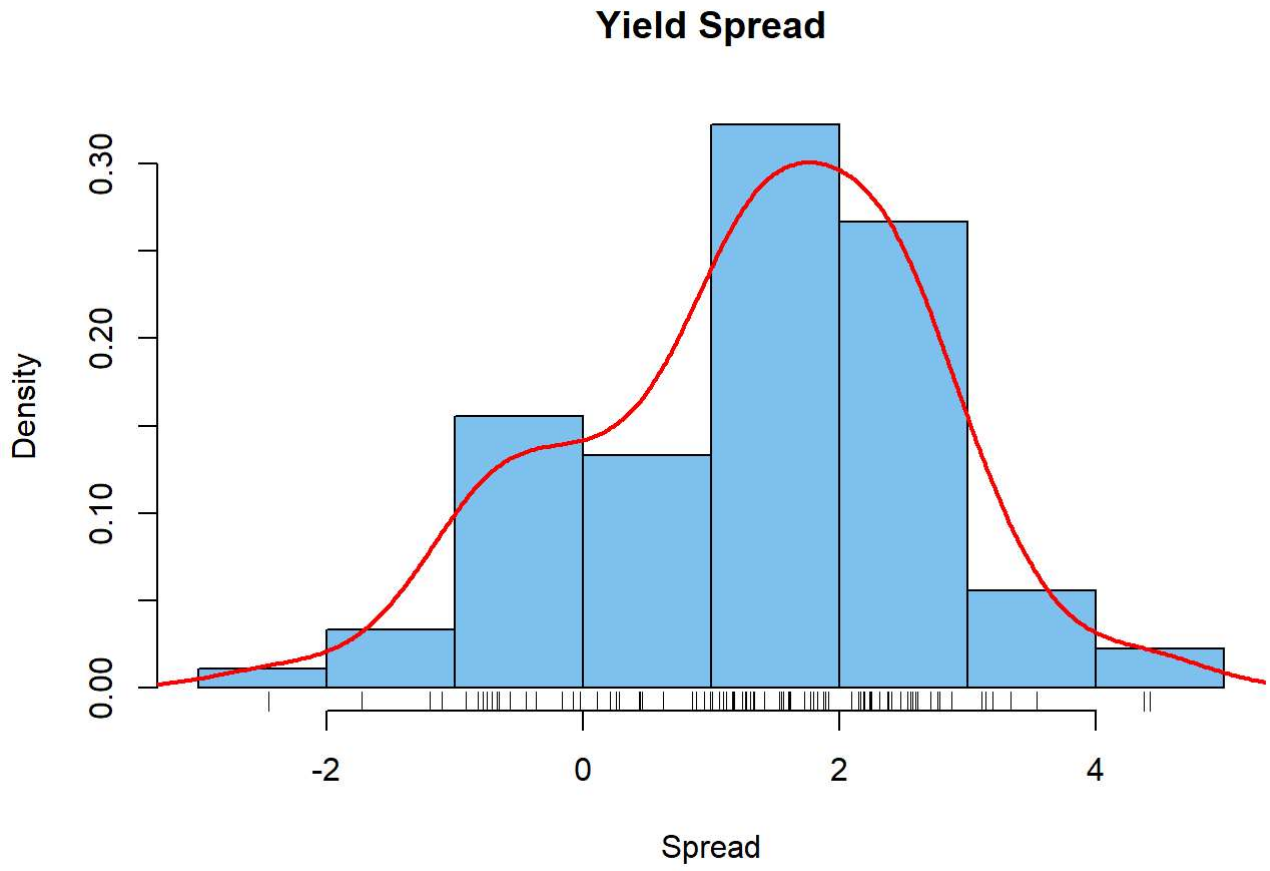
Problem 2c: Yield Spread

```
yield.spread.2 <- yield.spread[yield.spread$Date %in% recession.dates$dates, ]
head(yield.spread.2)
```

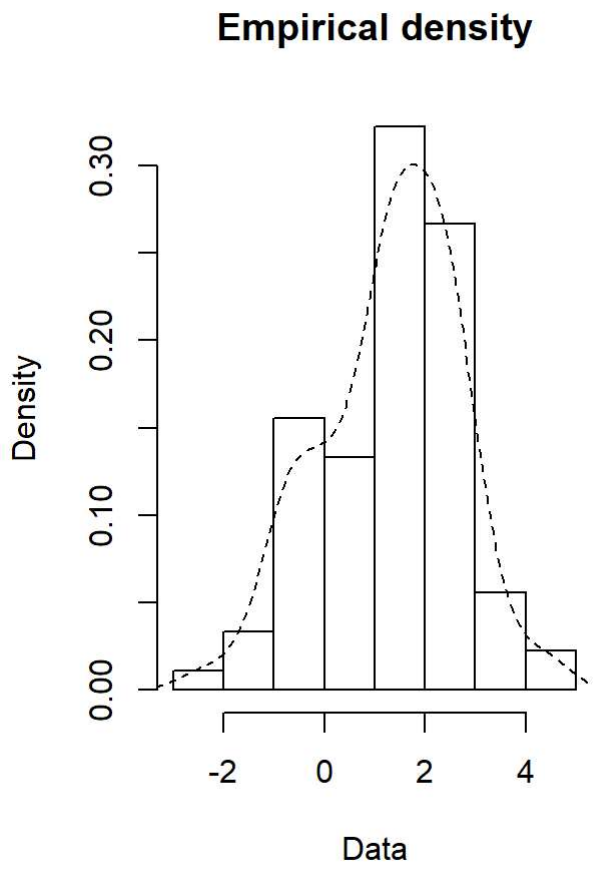
```
nrow(yield.spread.2)
```

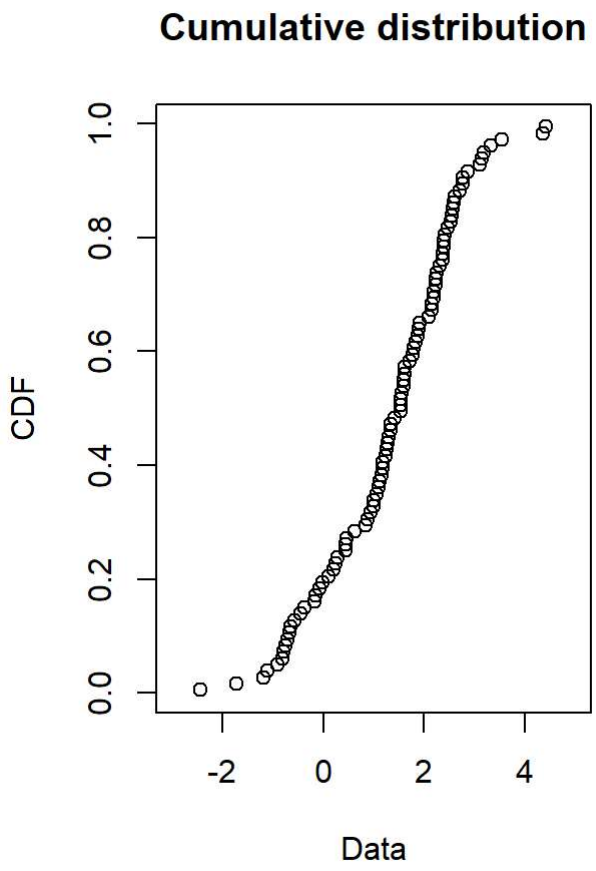
```
## [1] 90
```

```
ys2 <- yield.spread.2[, 2]
ys2 <- as.vector(ys2)
par(mfrow = c(1,1))
hist(ys2, breaks = "FD", prob = TRUE, main = "Yield Spread", col = "skyblue2",
     xlab = "Spread")
lines(density(ys2), col = "red", lwd = 2)
rug(ys2)
```

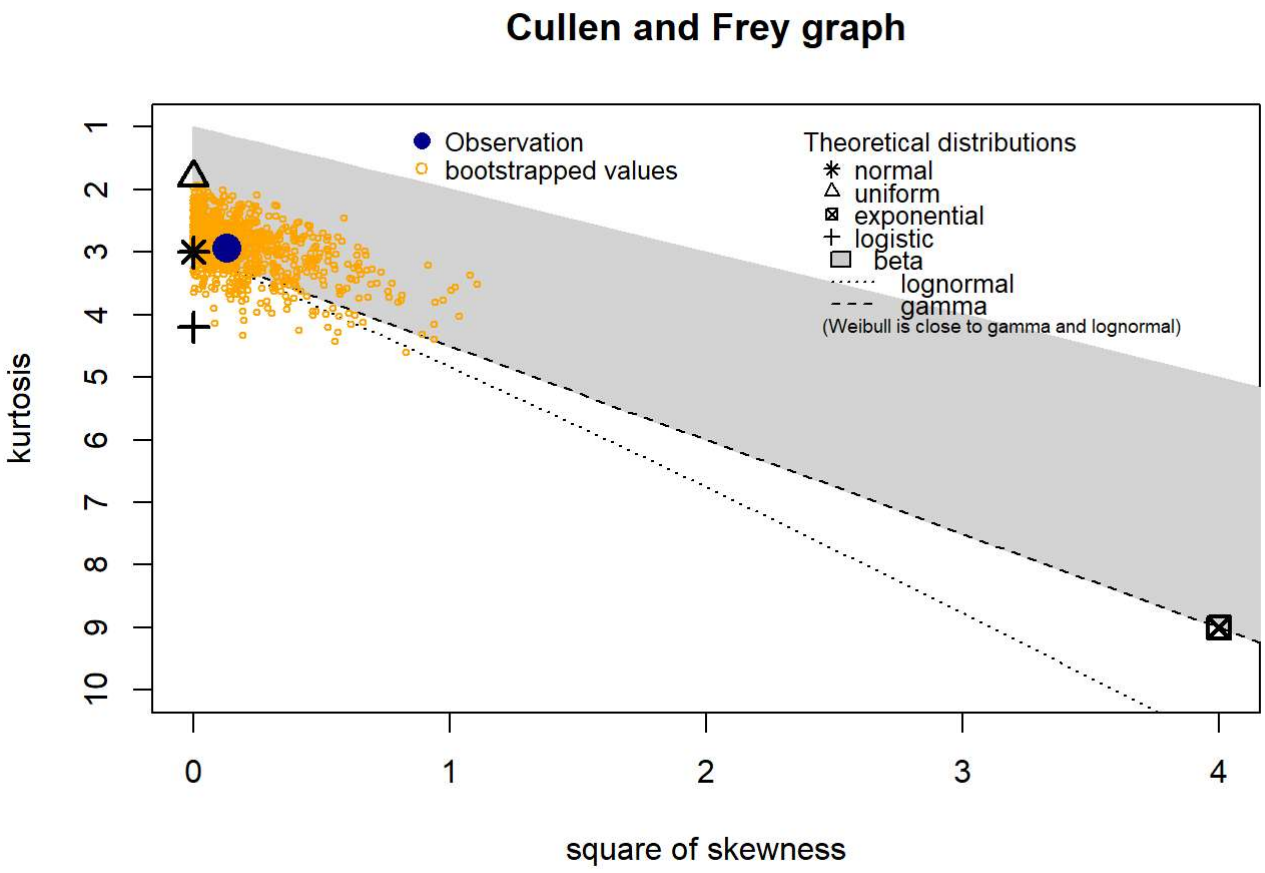



```
plotdist(ys2, histo = TRUE, demp = TRUE)
```






```
descdist(ys2, boot = 1000)
```

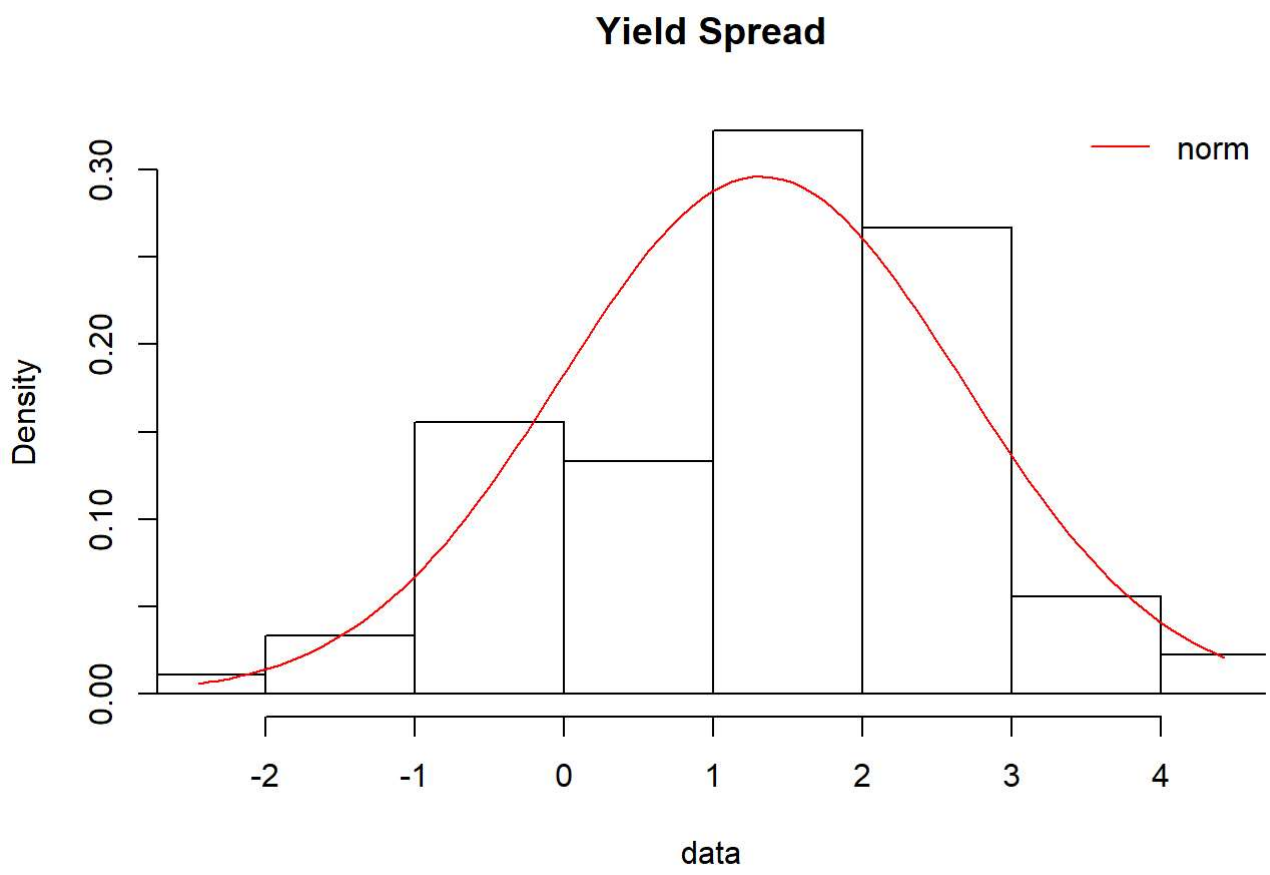



```
## summary statistics
## ------
## min:  -2.450952   max:  4.419048
## median:  1.532829
## mean:  1.318756
## estimated sd:  1.354278
## estimated skewness:  -0.3604789
## estimated kurtosis:  2.929936
```

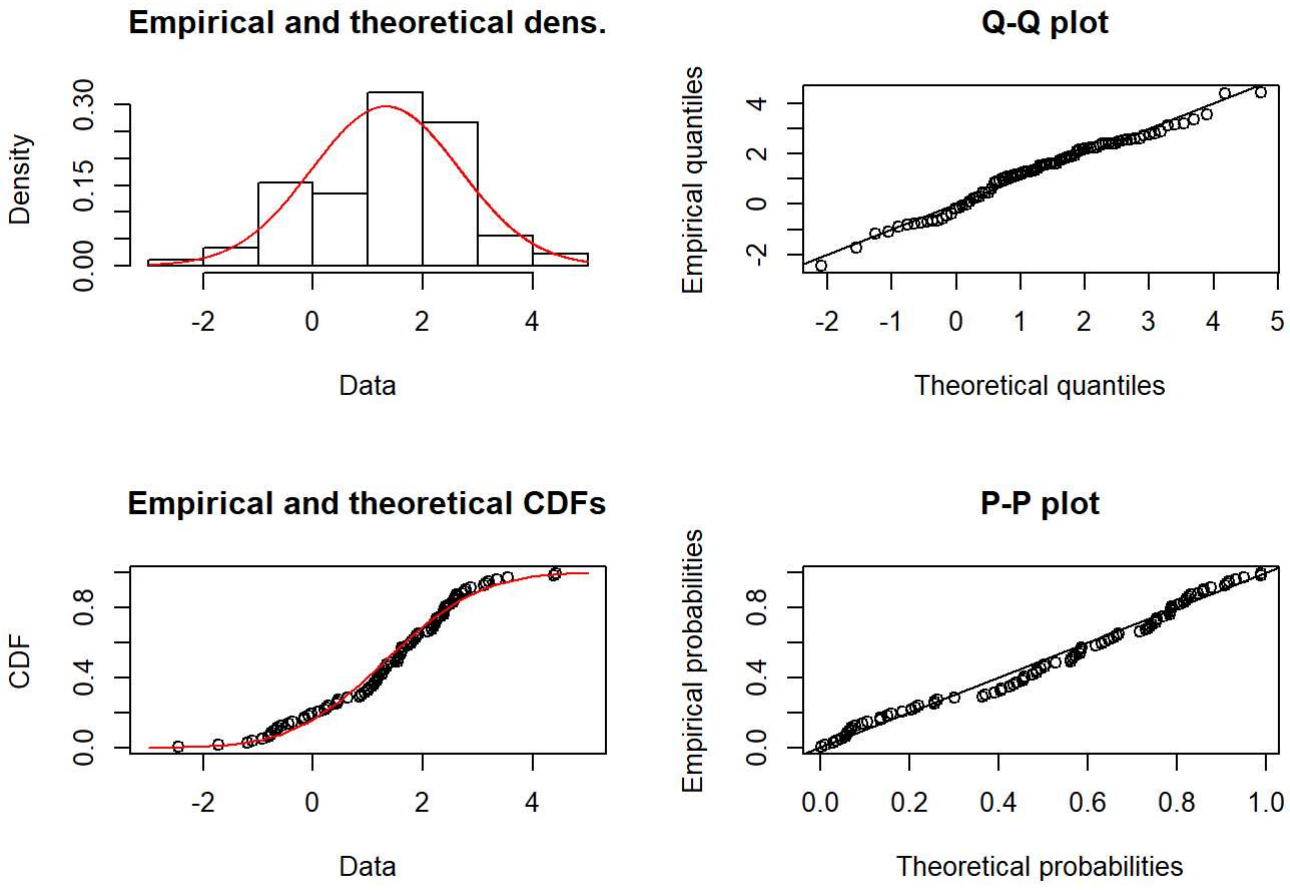
```
ys2.fitreturn <- fitdist(as.numeric(ys2), "norm")
ys2.fitreturn
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 1.318756  0.1419582
## sd   1.346734  0.1003794
```

```
denscomp(ys2.fitreturn, main = "Yield Spread")
```




```
plot(ys2.fitreturn)
```




```
ks.test(ys2, "pnorm", mean = ys2.fitreturn[["estimate"]][["mean"]],
        sd = ys2.fitreturn[["estimate"]][["sd"]])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  ys2
## D = 0.082582, p-value = 0.5436
## alternative hypothesis: two-sided
```

We test the data using a KS Test against the fitted normal distribution with the parameters estimated using MLE. The p-value is high, indicating that we fail to reject the null and the data comes from the fitted distribution. Therefore, we determine that the data follows a normal distribution. Based on the Maximum Likelihood Estimation for the parameters the data follows a normal distribution with mean = 1.3187 and sd = 1.3467

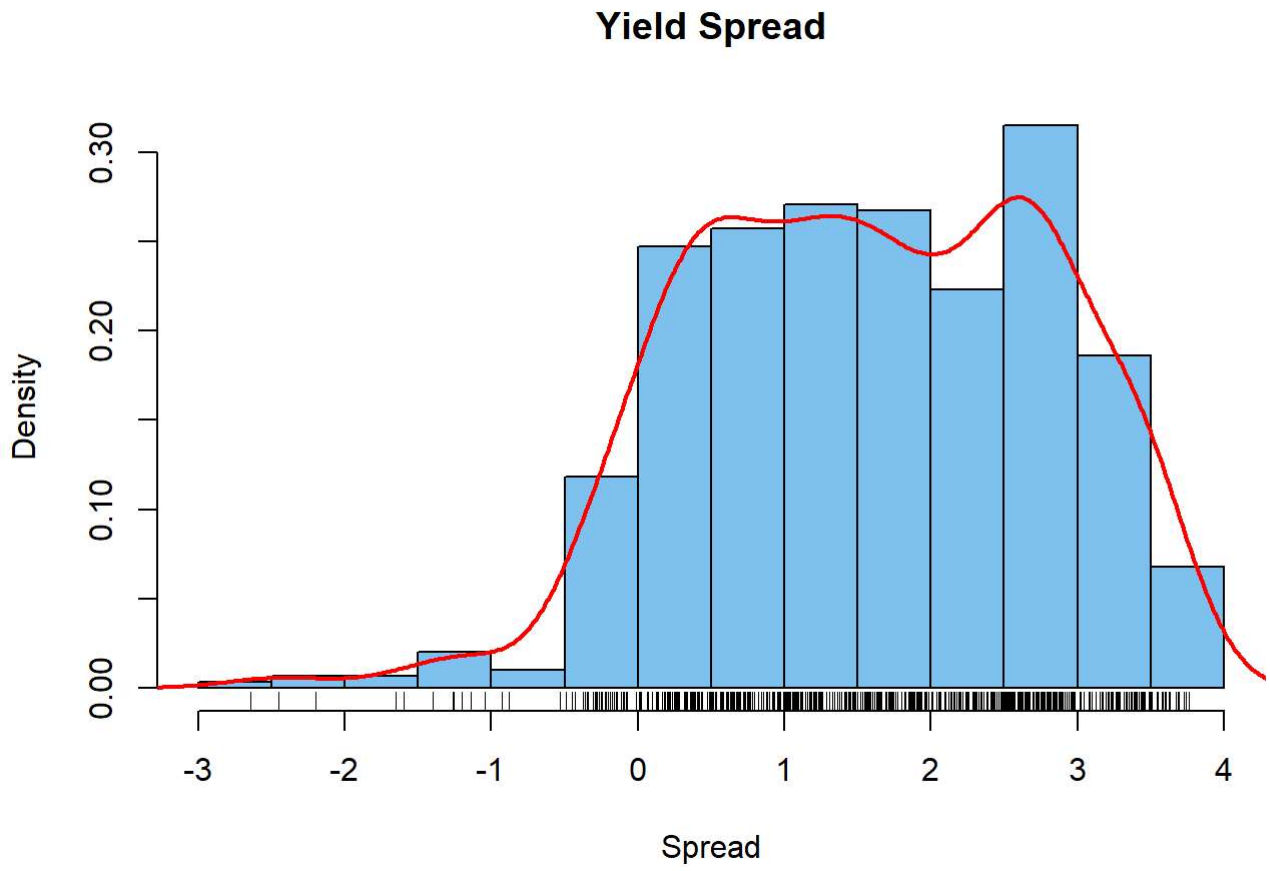
Problem 2d: Yield Spread

```
yield.spread.3 <- yield.spread[!(yield.spread$Date %in% recession.dates$dates), ]
head(yield.spread.3)
```

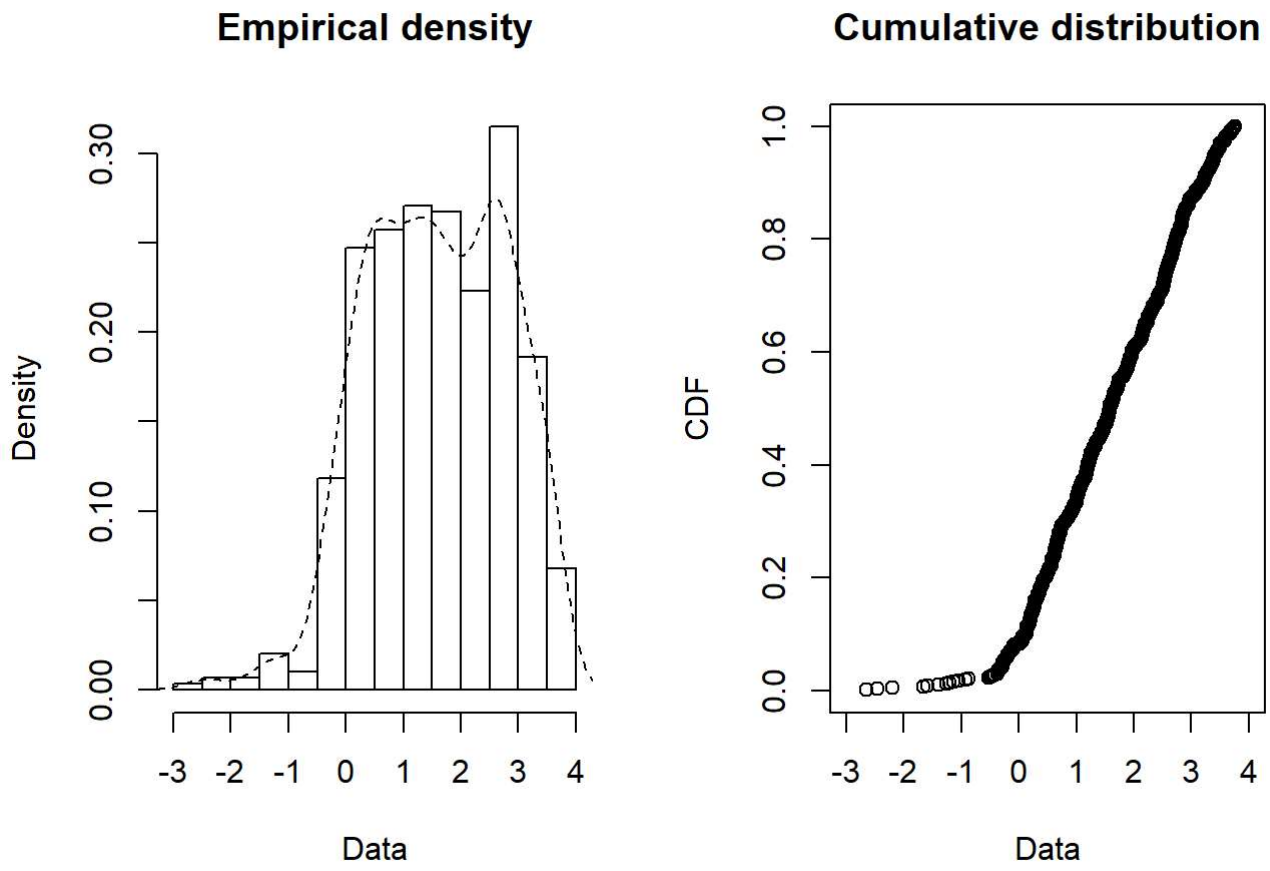
```
nrow(yield.spread.3)
```

```
## [1] 591
```

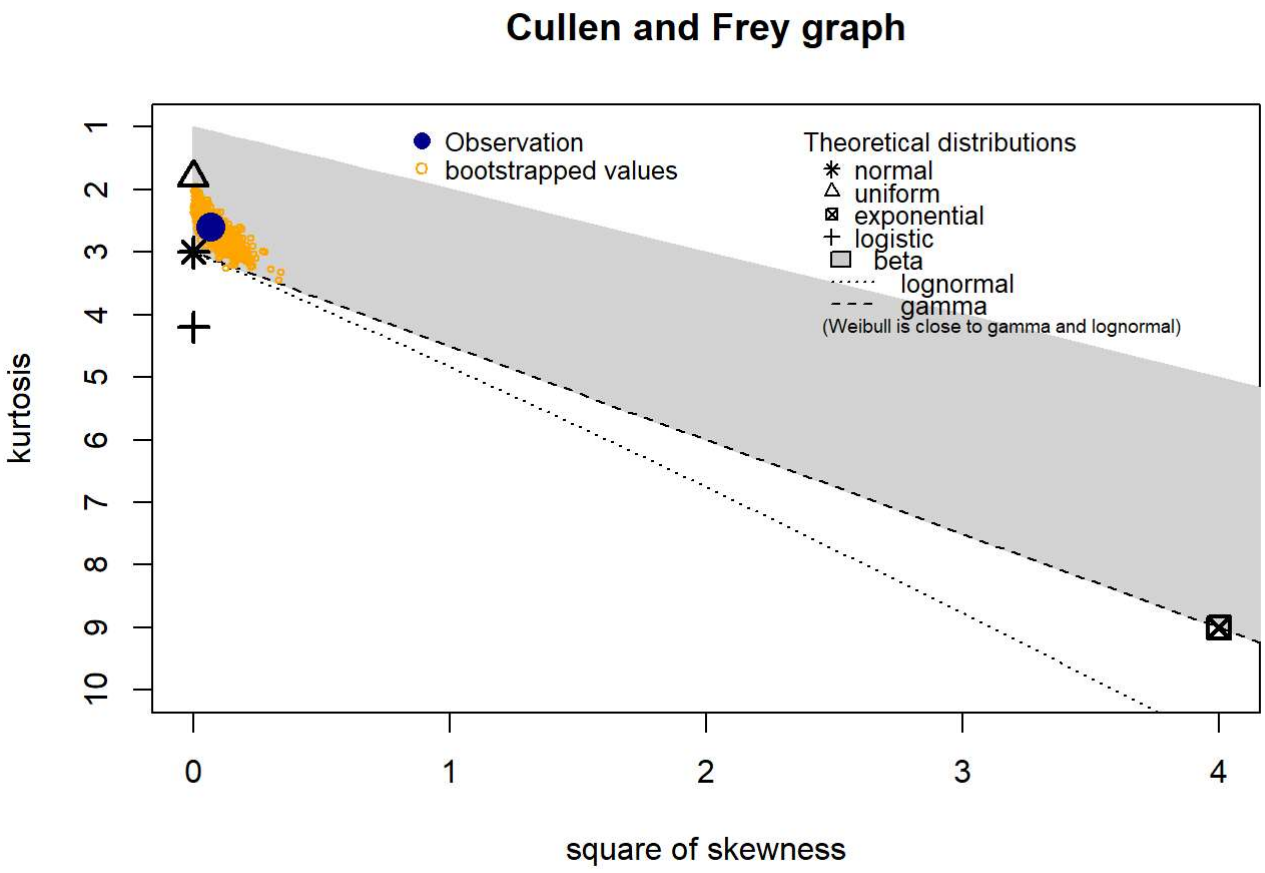
```
ys3 <- yield.spread.3[, 2]
ys3 <- as.vector(ys3)
par(mfrow = c(1,1))
hist(ys3, breaks = "FD", prob = TRUE, main = "Yield Spread", col = "skyblue2",
     xlab = "Spread")
lines(density(ys3), col = "red", lwd = 2)
rug(ys3)
```




```
plotdist(ys3, histo = TRUE, demp = TRUE)
```




```
descdist(ys3, boot = 1000)
```

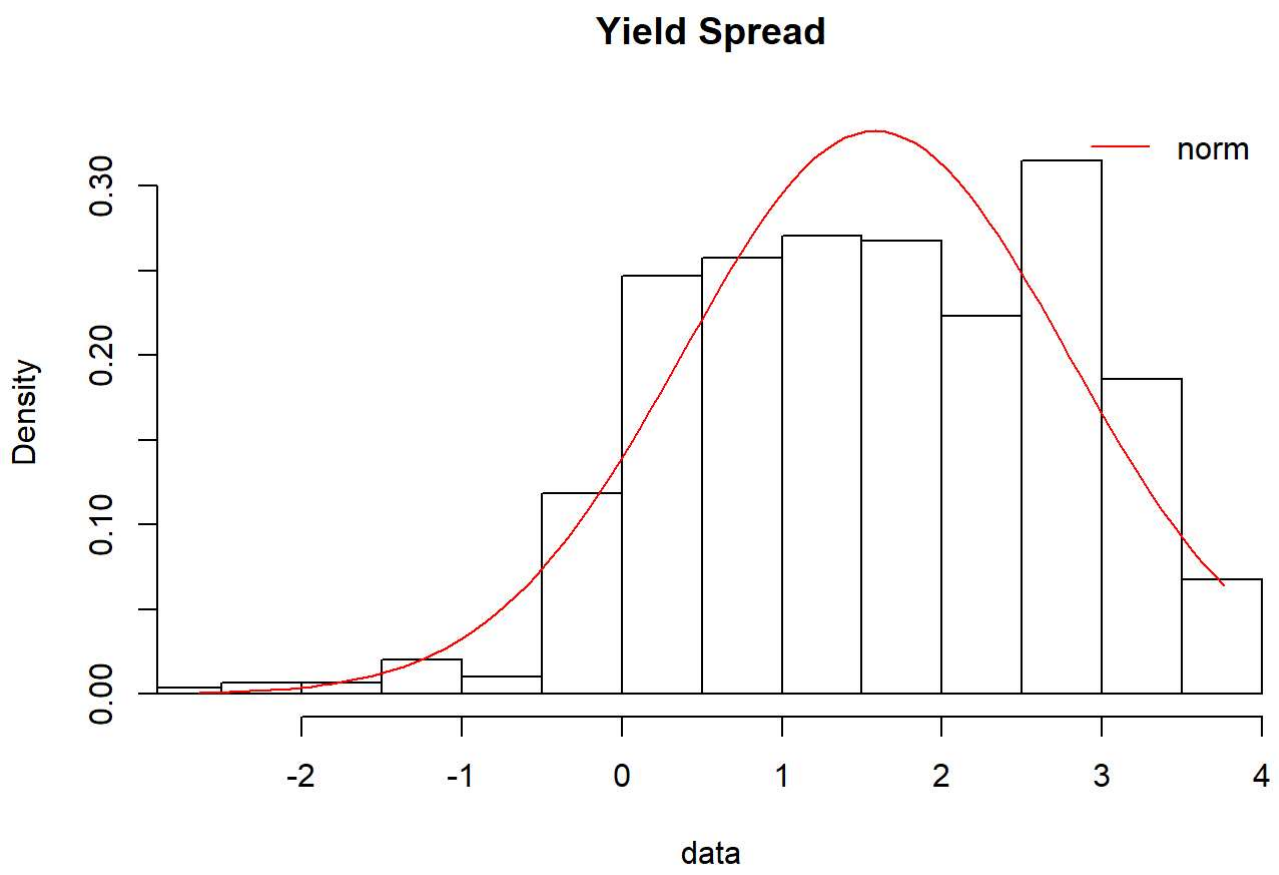



```
## summary statistics
## ------
## min:  -2.645909   max:  3.762
## median:  1.583333
## mean:  1.582331
## estimated sd:  1.201376
## estimated skewness:  -0.261962
## estimated kurtosis:  2.593887
```

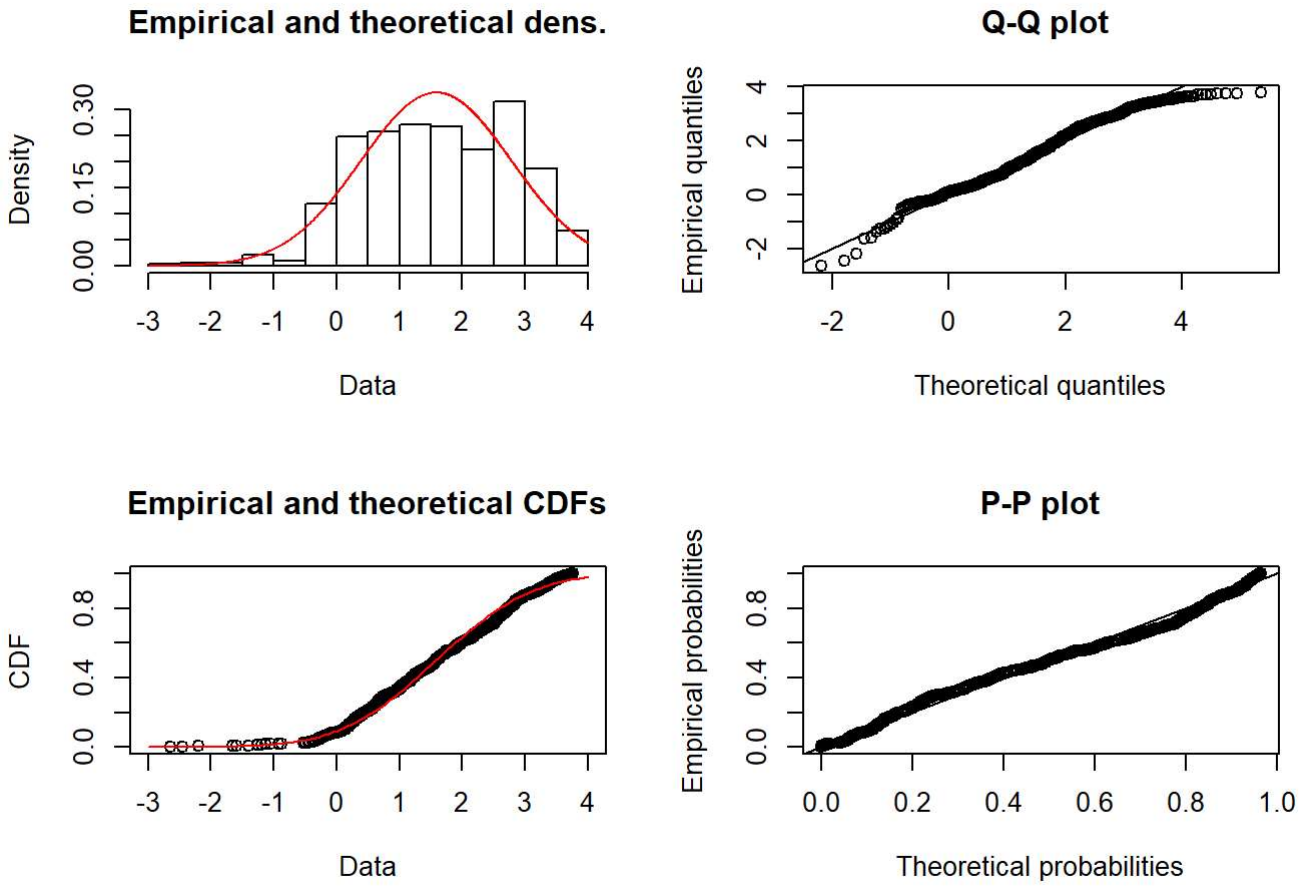
```
ys3.fitreturn <- fitdist(as.numeric(ys3), "norm")
ys3.fitreturn
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 1.582331 0.04937617
## sd   1.200359 0.03491412
```

```
denscomp(ys3.fitreturn, main = "Yield Spread")
```




```
plot(ys3.fitreturn)
```




```
ks.test(ys3, "pnorm", mean = ys3.fitreturn[["estimate"]][["mean"]],
        sd = ys3.fitreturn[["estimate"]][["sd"]])
```

```
## Warning in ks.test(ys3, "pnorm", mean = ys3.fitreturn[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  ys3
## D = 0.066573, p-value = 0.01062
## alternative hypothesis: two-sided
```

We determine that the data follows a normal distribution. Based on the Maximum Likelihood Estimation for the parameters the data follows a normal distribution with mean = 1.5823 and sd = 1.2003

Problem 2e: Yield Spread

Based on our findings, when the data is subsetted based on recession time periods, the samples all follow a normal distribution. However, Maximum Likelihood Estimation for the parameters shows that each sample follows a normal distribution with different parameters for the mean and sd. It appears that the sample with all the data and the sample with only the points generated during the non-recession periods have parameters that are very close.

Problem 2f: Yield Spread

```
ks.test(ys, ys2)
```

```
## Warning in ks.test(ys, ys2): p-value will be approximate in the presence of
## ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  ys and ys2
## D = 0.1071, p-value = 0.3216
## alternative hypothesis: two-sided
```

```
ks.test(ys, ys3)
```

```
## Warning in ks.test(ys, ys3): p-value will be approximate in the presence of
## ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  ys and ys3
## D = 0.016309, p-value = 1
## alternative hypothesis: two-sided
```

```
ks.test(ys2, ys3)
```

```
## Warning in ks.test(ys2, ys3): p-value will be approximate in the presence
## of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  ys2 and ys3
## D = 0.12341, p-value = 0.1851
## alternative hypothesis: two-sided
```

The KS Tests confirm my findings in part (e). The three samples all come from a normal distribution with similar parameters.

Problem 2a: Three Month Treasury Rate

```
three.mo <- read.csv("3 Month Fred.csv", header = TRUE)
date <- as.Date(three.mo$DATE, format = "%Y-%m-%d")
three.mo <- cbind(date, three.mo)
three.mo <- three.mo[, -2]
names(three.mo) <- paste(c("Date", "Three.month"))
head(three.mo, n = 3)
```

```
tail(three.mo, n = 3)
```

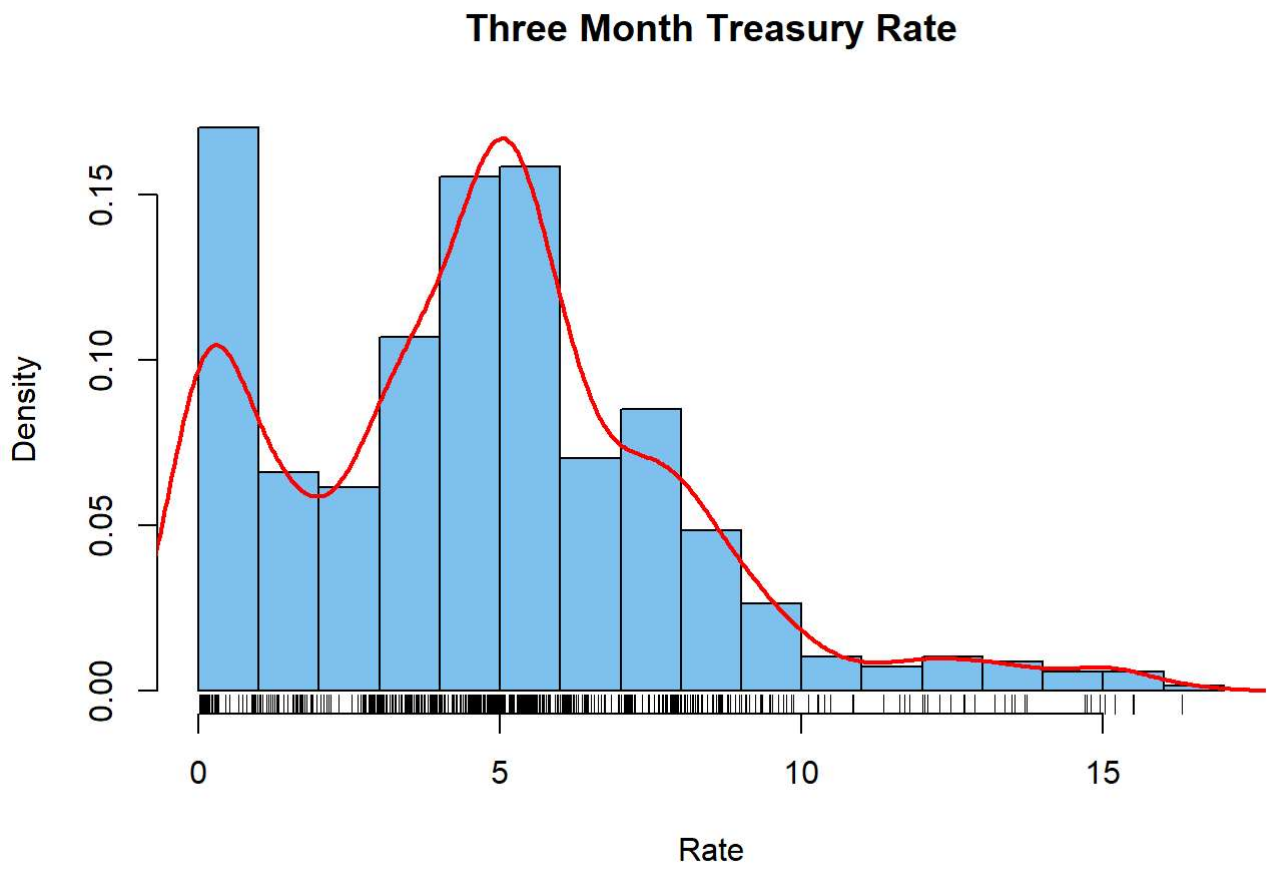
```
nrow(three.mo)
```

```
## [1] 681
```

```
tmo <- three.mo[, 2]
tmo <- as.vector(tmo)
summary(tmo)
```

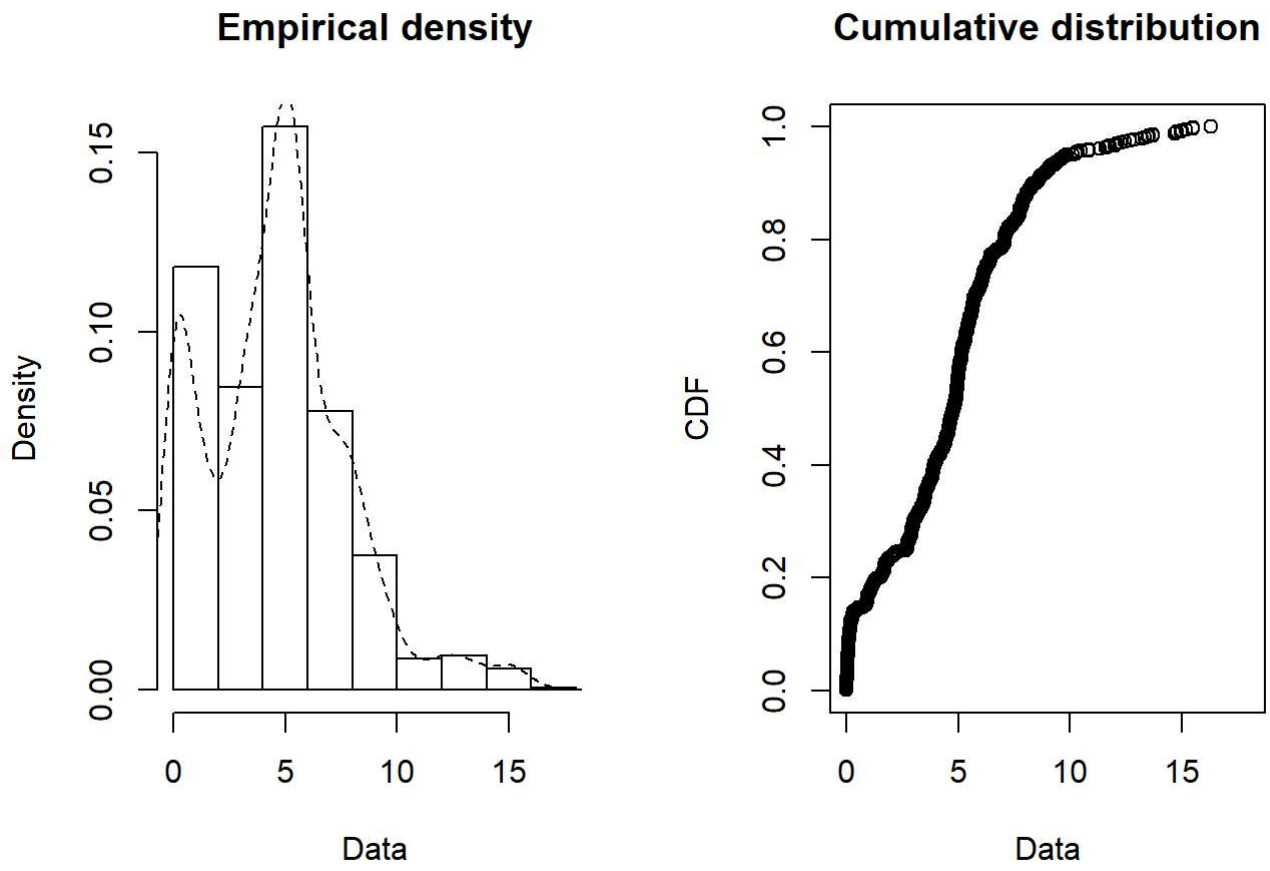
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.690   4.800   4.656   6.260  16.300
```

```
par(mfrow = c(1,1))
hist(tmo, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Three Month Treasury Rate",
     xlab = "Rate")
lines(density(tmo), col = "red", lwd = 2)
rug(tmo)
```

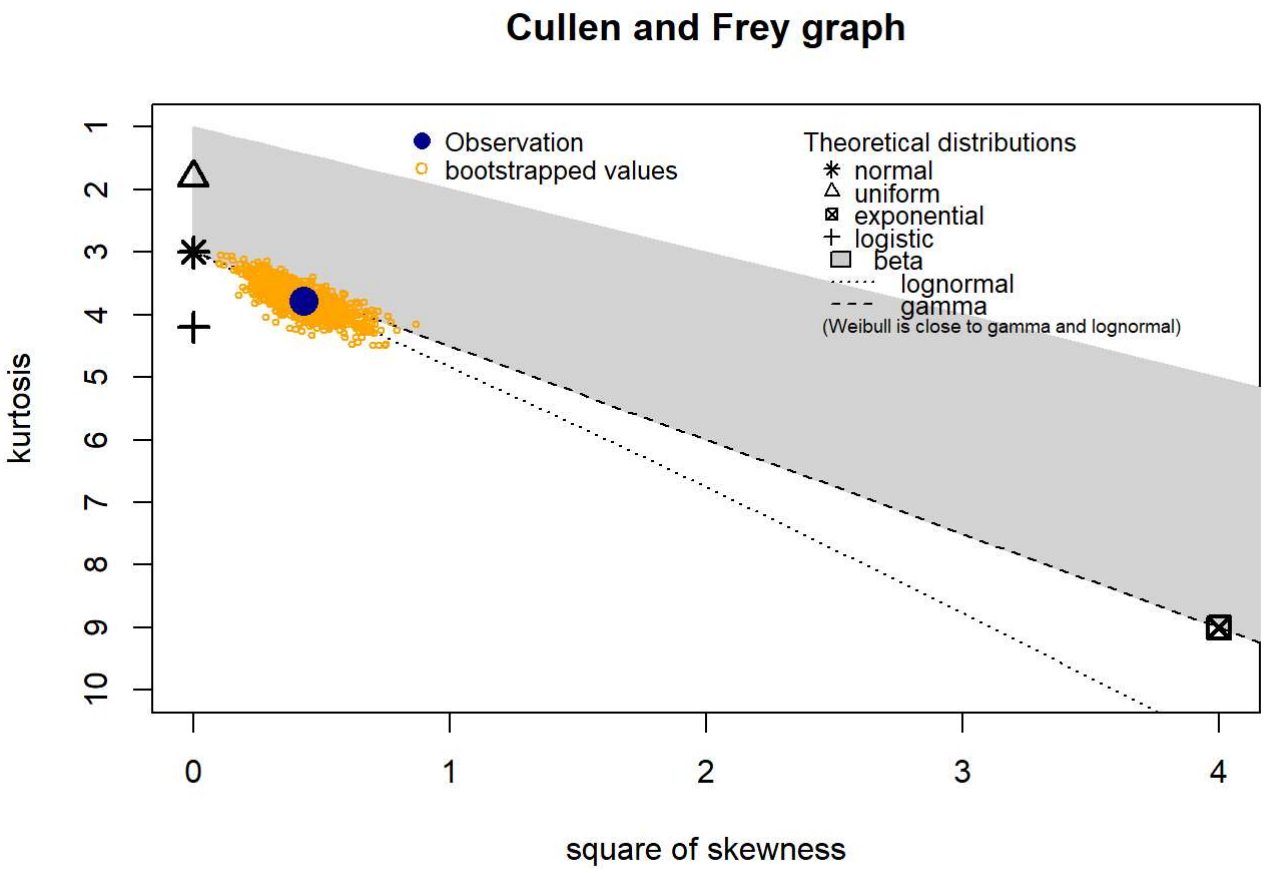


Problem 2b: Three Month Treasury Rate

```
plotdist(tmo, histo = TRUE, demp = TRUE)
```




```
descdist(tmo, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.01   max:  16.3
## median:  4.8
## mean:  4.656021
## estimated sd:  3.200114
## estimated skewness:  0.6550402
## estimated kurtosis:  3.776337
```

```
tmo.fit.lognormal <- fitdist(as.numeric(tmo), "lnorm")
tmo.fit.lognormal
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 0.9406118 0.06119101
## sdlog   1.5968392 0.04326850
```

```
tmo.fit.gamma <- fitdist(as.numeric(tmo), "gamma")
tmo.fit.gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.9697381 0.04614507
## rate  0.2082669 0.01280181
```

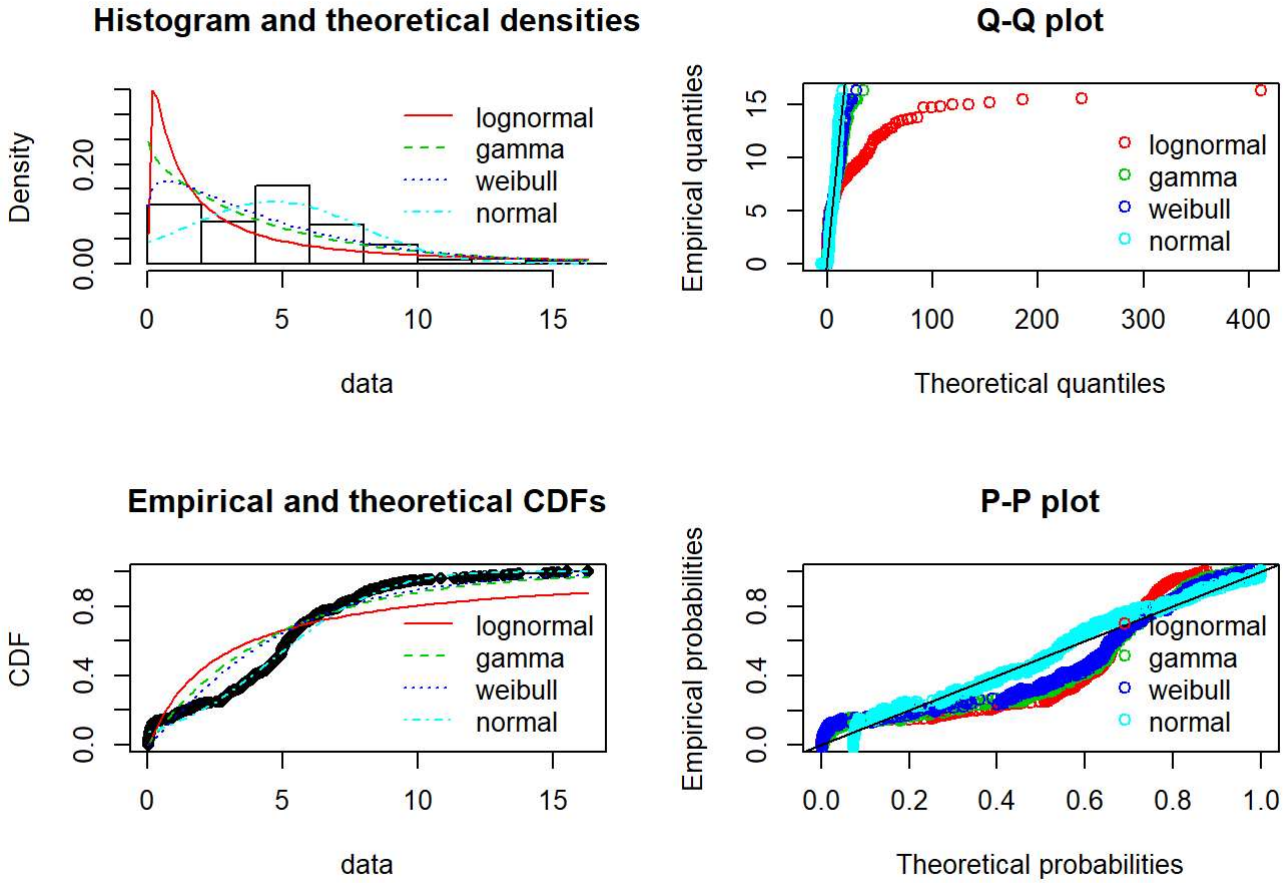
```
tmo.fit.weibull <- fitdist(as.numeric(tmo), "weibull")
tmo.fit.weibull
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.113108 0.03721319
## scale 4.793908 0.17069908
```

```
tmo.fit.norm <- fitdist(as.numeric(tmo), "norm")
tmo.fit.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 4.656021 0.12253858
## sd   3.197764 0.08664782
```

```
par(mfrow=c(2,2))
plot.legend <- c("lognormal", "gamma", "weibull", "normal")
denscomp(list(tmo.fit.lognormal, tmo.fit.gamma, tmo.fit.weibull, tmo.fit.norm),
         legendtext = plot.legend)
qqcomp(list(tmo.fit.lognormal, tmo.fit.gamma, tmo.fit.weibull, tmo.fit.norm),
       legendtext = plot.legend)
cdfcomp(list(tmo.fit.lognormal, tmo.fit.gamma, tmo.fit.weibull, tmo.fit.norm),
        legendtext = plot.legend)
ppcomp(list(tmo.fit.lognormal, tmo.fit.gamma, tmo.fit.weibull, tmo.fit.norm),
       legendtext = plot.legend)
```



It appears that the normal distribution correctly describes the center and the tails of the distribution of the data more than the others.

```
ks.test(tmo, "pgamma", shape = tmo.fit.gamma[["estimate"]][["shape"]],
        rate = tmo.fit.gamma[["estimate"]][["rate"]])
```

```
## Warning in ks.test(tmo, "pgamma", shape = tmo.fit.gamma[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo
## D = 0.19725, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(tmo, "pweibull", shape = tmo.fit.weibull[["estimate"]][["shape"]],
        scale = tmo.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(tmo, "pweibull", shape = tmo.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo
## D = 0.17007, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(tmo, "pnorm", mean = tmo.fit.norm[["estimate"]][["mean"]],
        sd = tmo.fit.norm[["estimate"]][["sd"]])
```

```
## Warning in ks.test(tmo, "pnorm", mean = tmo.fit.norm[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo
## D = 0.073126, p-value = 0.001374
## alternative hypothesis: two-sided
```

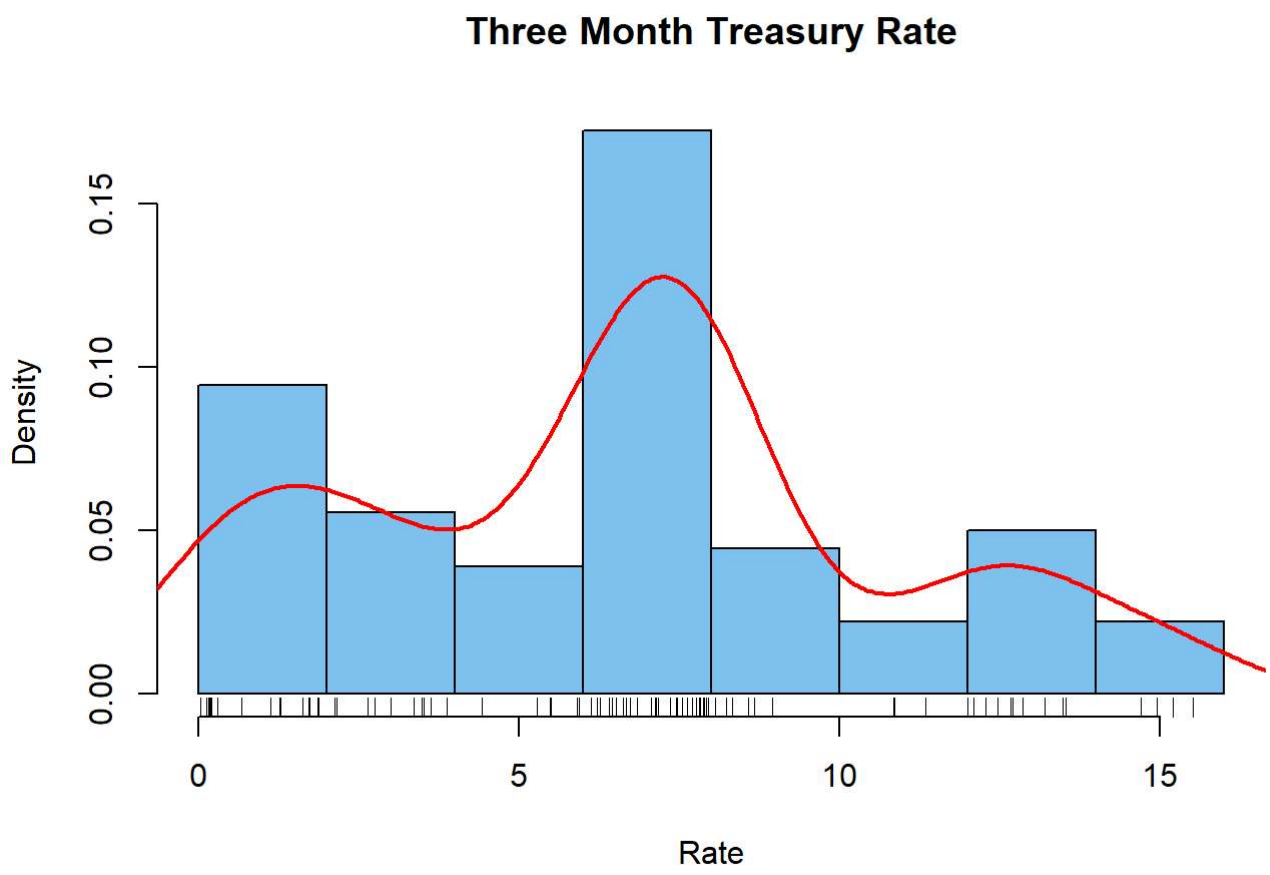
The p-value for the KS Test of the data against the fitted normal distribution, using the estimated parameters from MLE, is the highest. This indicates that it is the closest fitted distribution to the data compared to the others. We determine the normal to be the best fitted disrtibution to describe the data. The Maximum Likelihood Estimation of the parameters when the data is fitted to a weibull distribution is a weibull distribution with mean = 4.65 and sd = 3.19

Problem 2c: Three Month Treasury Rate

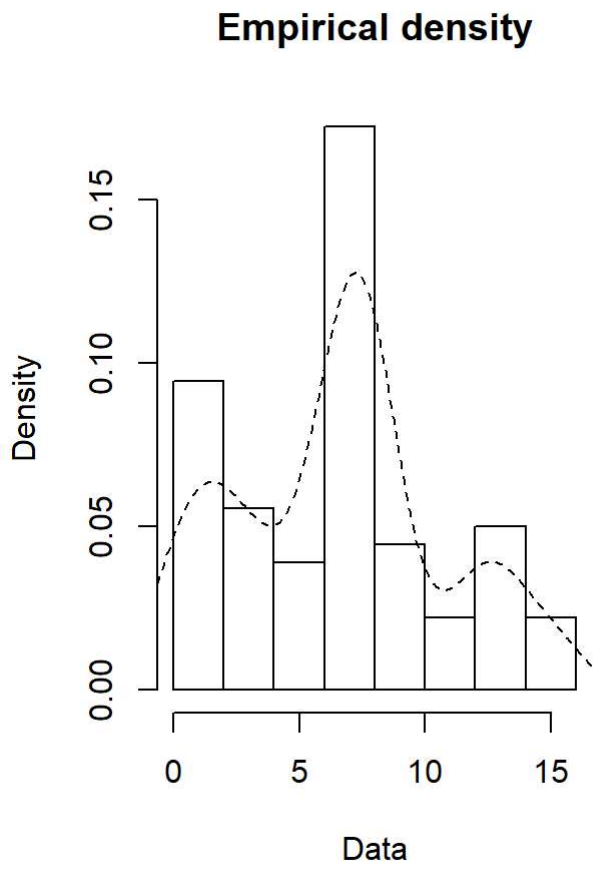
```
three.mo2 <- three.mo[three.mo$Date %in% recession.dates$dates, ]
nrow(three.mo2)
```

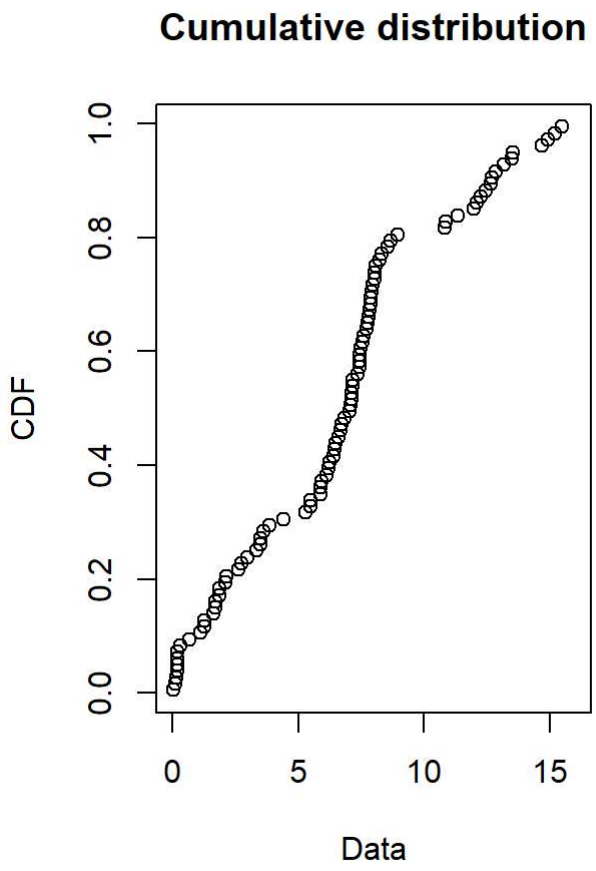
```
## [1] 90
```

```
tmo2 <- three.mo2[, 2]
tmo2 <- as.vector(tmo2)
par(mfrow=c(1,1))
hist(tmo2, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Three Month Treasury Rate",
     xlab = "Rate")
lines(density(tmo2), col = "red", lwd = 2)
rug(tmo2)
```

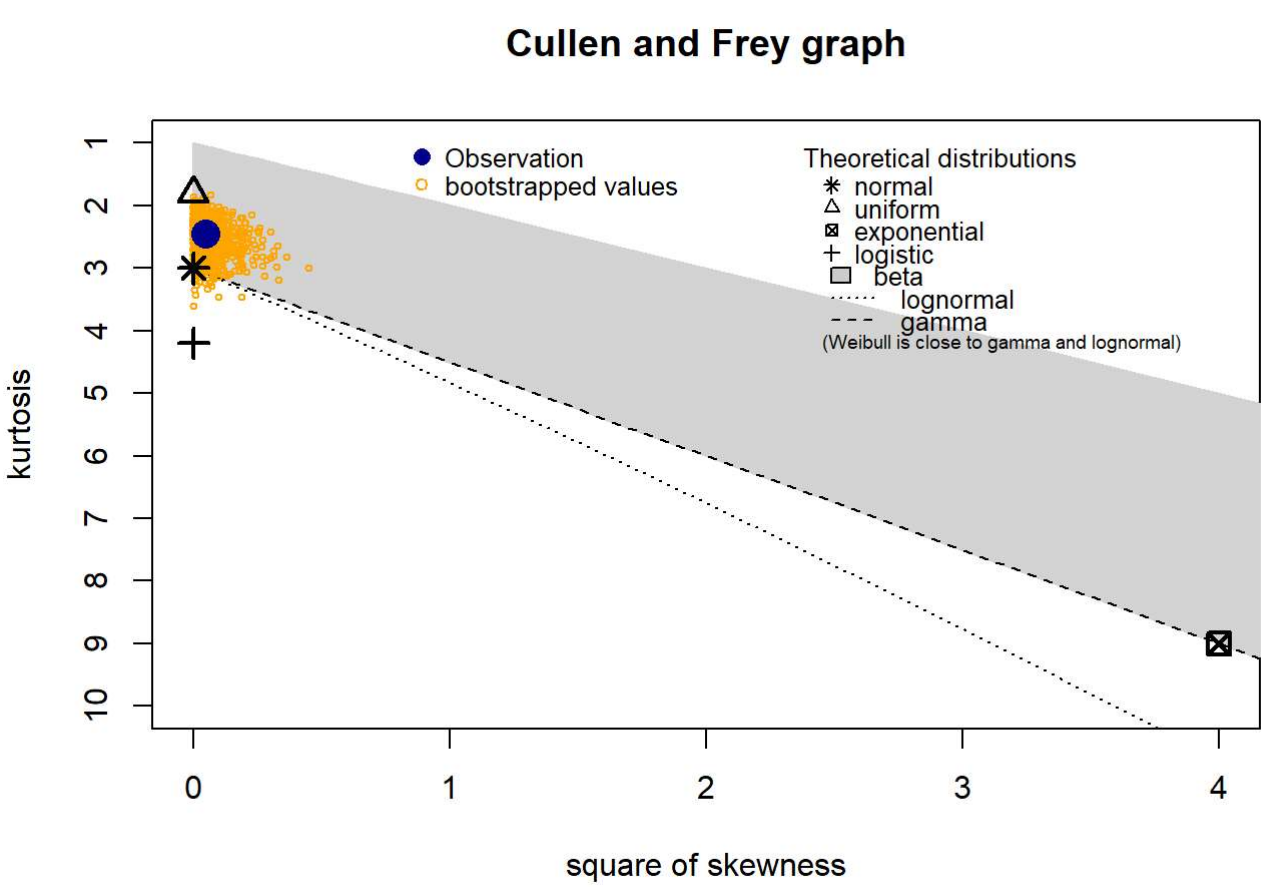



```
plotdist(tmo2, histo = TRUE, demp = TRUE)
```






```
descdist(tmo2, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.03   max:  15.51
## median:  7.065
## mean:  6.598889
## estimated sd:  4.079317
## estimated skewness:  0.2166997
## estimated kurtosis:  2.456742
```

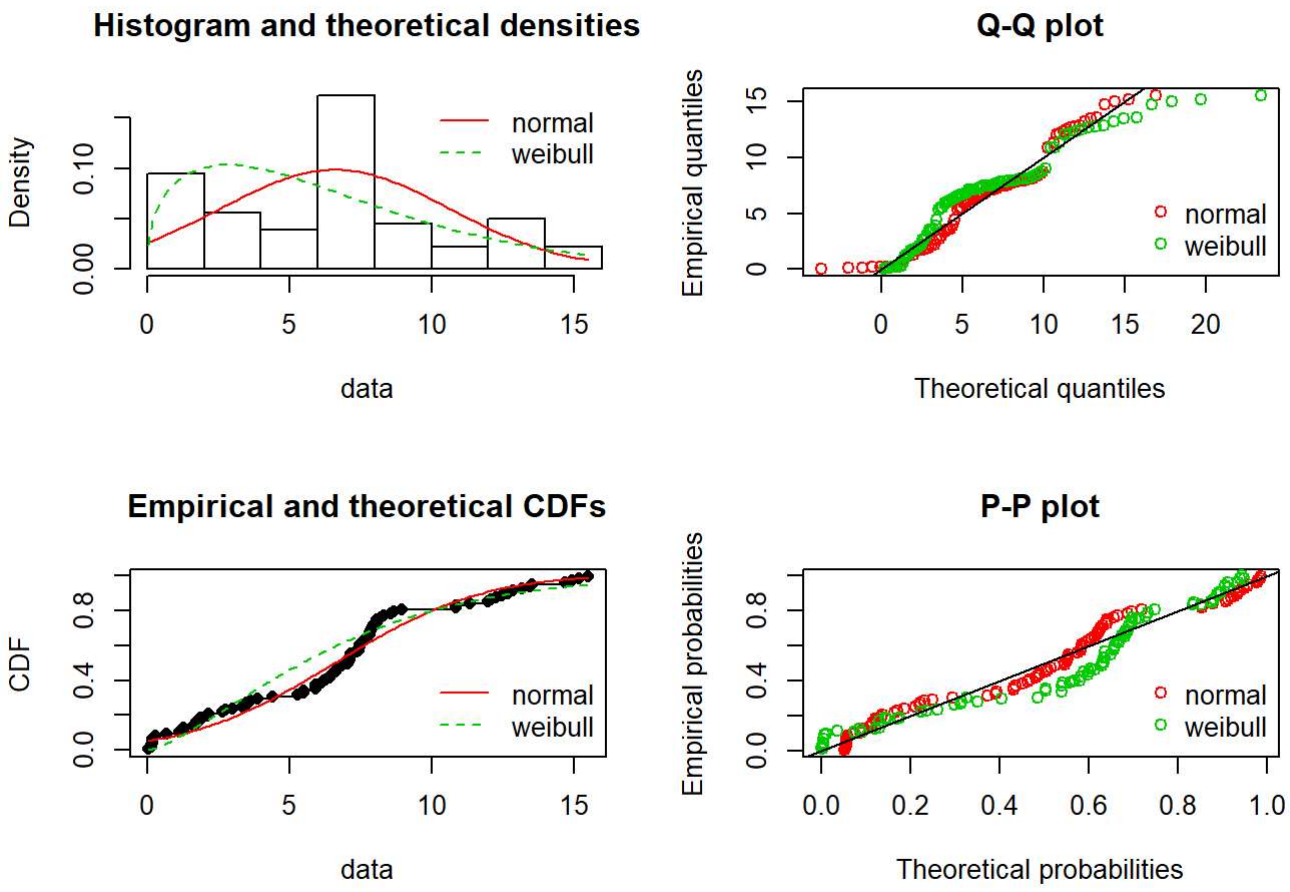
```
tmo2.fit.norm <- fitdist(as.numeric(tmo2), "norm")
tmo2.fit.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 6.598889  0.4276022
## sd   4.056590  0.3023603
```

```
tmo2.fit.weibull <- fitdist(as.numeric(tmo2), "weibull")
tmo2.fit.weibull
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.383511  0.1254615
## scale 7.101940  0.5610741
```

```
par(mfrow=c(2,2))
plot.legend <- c("normal", "weibull")
denscomp(list(tmo2.fit.norm, tmo2.fit.weibull), legendtext = plot.legend)
qqcomp(list(tmo2.fit.norm, tmo2.fit.weibull), legendtext = plot.legend)
cdfcomp(list(tmo2.fit.norm, tmo2.fit.weibull), legendtext = plot.legend)
ppcomp(list(tmo2.fit.norm, tmo2.fit.weibull), legendtext = plot.legend)
```




```
ks.test(tmo2, "pnorm", mean = tmo2.fit.norm[["estimate"]][["mean"]],
        sd = tmo2.fit.norm[["estimate"]][["sd"]])
```

```
## Warning in ks.test(tmo2, "pnorm", mean = tmo2.fit.norm[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo2
## D = 0.11399, p-value = 0.1927
## alternative hypothesis: two-sided
```

```
ks.test(tmo2, "pweibull", shape = tmo2.fit.weibull[["estimate"]][["shape"]],
        scale = tmo2.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(tmo2, "pweibull", shape = tmo2.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo2
## D = 0.19511, p-value = 0.002115
## alternative hypothesis: two-sided
```

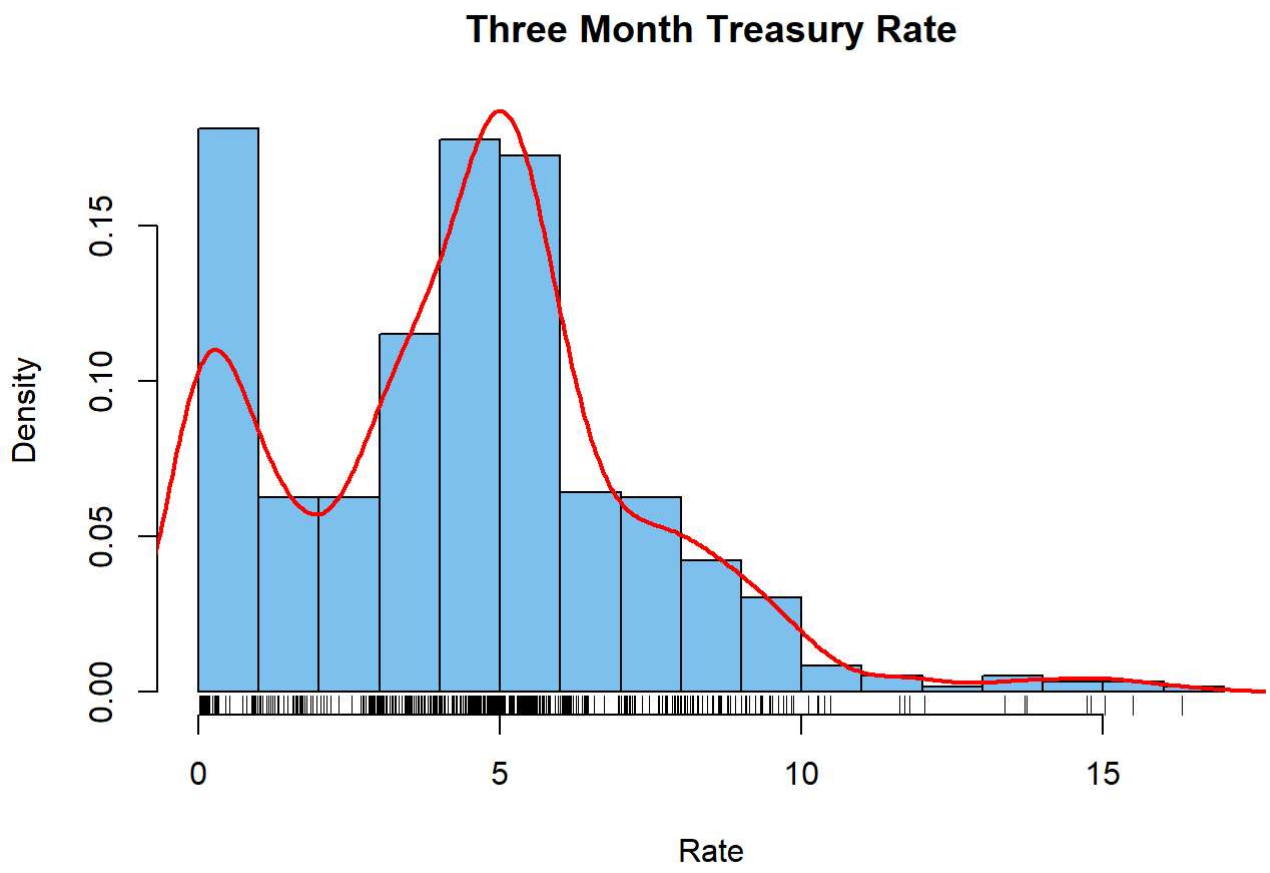
Using the KS Test, the p-value for the normal data against the normal distribution is high, indicating that we fail to reject the null that the data comes from a normal distribution with the parameters estimated using MLE. We determine the normal distribution to be the best fitted disrtibution to describe the data for points generated during recession periods. The Maximum Likelihood Estimation of the parameters when the data is fitted to a weibull distribution is a normal distribution with mean = 6.5988 and sd = 4.0565

Problem 2d: Three Month Treasury Rate

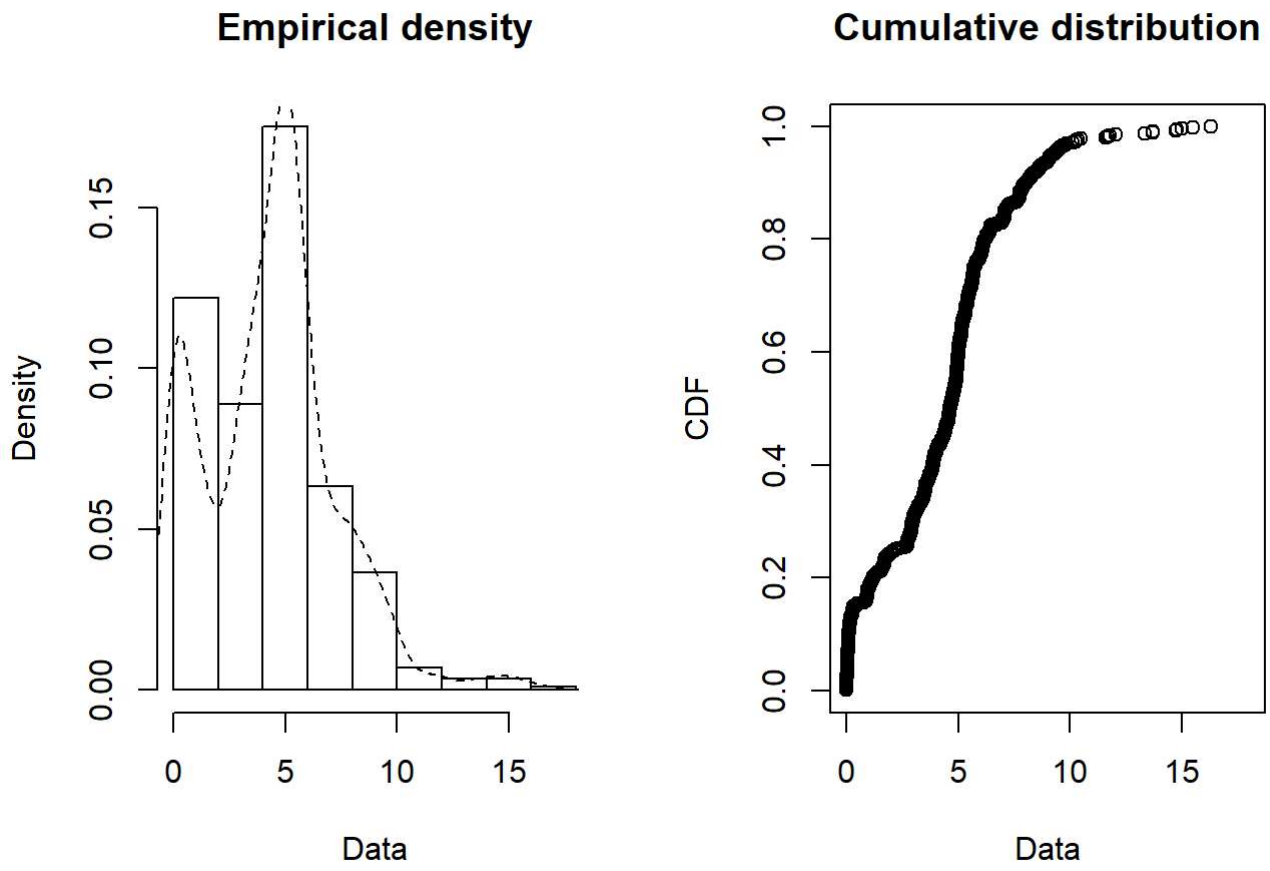
```
three.mo3 <- three.mo[!(three.mo$Date %in% recession.dates$dates), ]
nrow(three.mo3)
```

```
## [1] 591
```

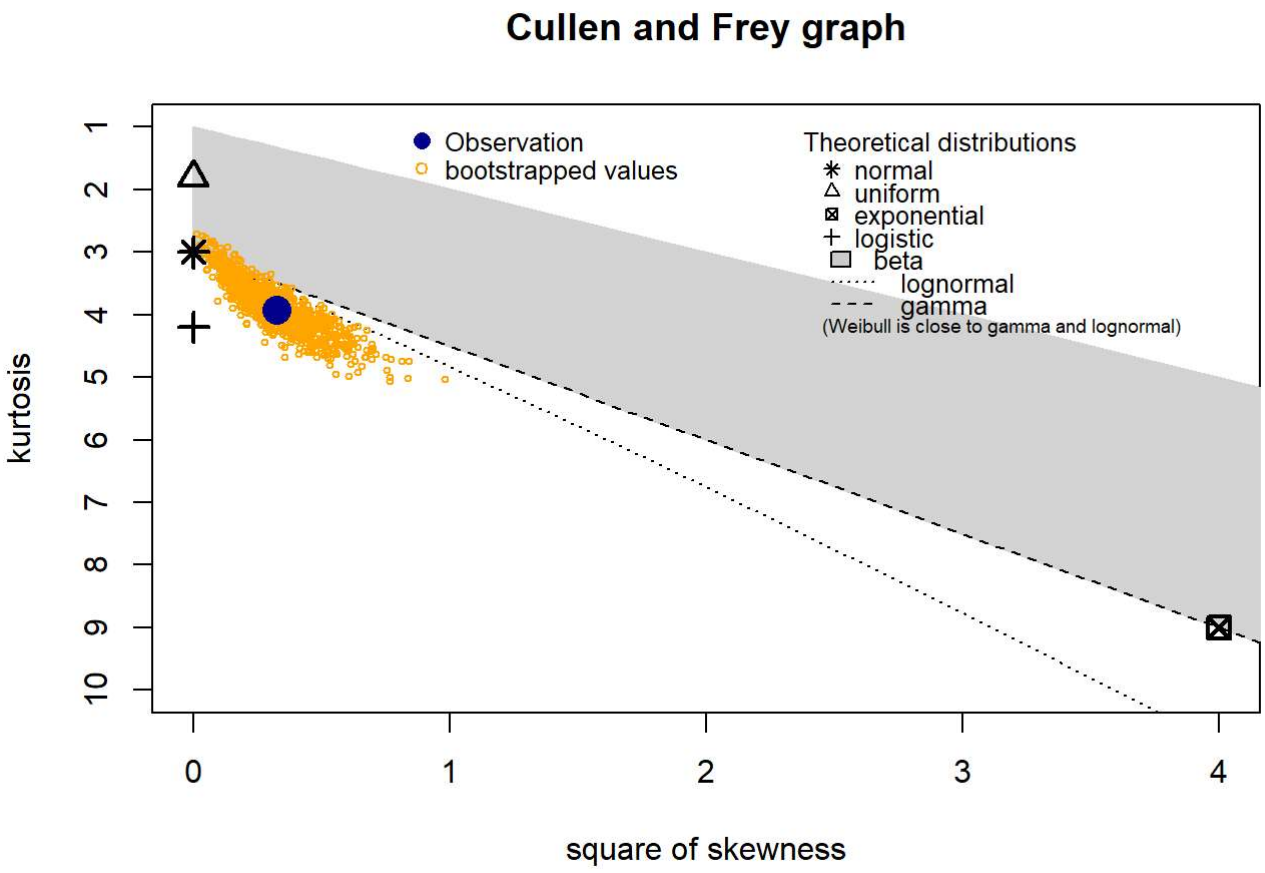
```
tmo3 <- three.mo3[, 2]
tmo3 <- as.vector(tmo3)
par(mfrow=c(1,1))
hist(tmo3, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Three Month Treasury Rate",
     xlab = "Rate")
lines(density(tmo3), col = "red", lwd = 2)
rug(tmo3)
```




```
plotdist(tmo3, histo = TRUE, demp = TRUE)
```




```
descdist(tmo3, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.01   max:  16.3
## median:  4.62
## mean:  4.360152
## estimated sd:  2.937543
## estimated skewness:  0.571366
## estimated kurtosis:  3.928509
```

```
tmo3.fit.lognormal <- fitdist(as.numeric(tmo3), "lnorm")
tmo3.fit.lognormal
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 0.8602302 0.06697675
## sdlog   1.6282376 0.04735963
```

```
tmo3.fit.gamma <- fitdist(as.numeric(tmo3), "gamma")
tmo3.fit.gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.9485657  0.0483571
## rate  0.2175611  0.0144025
```

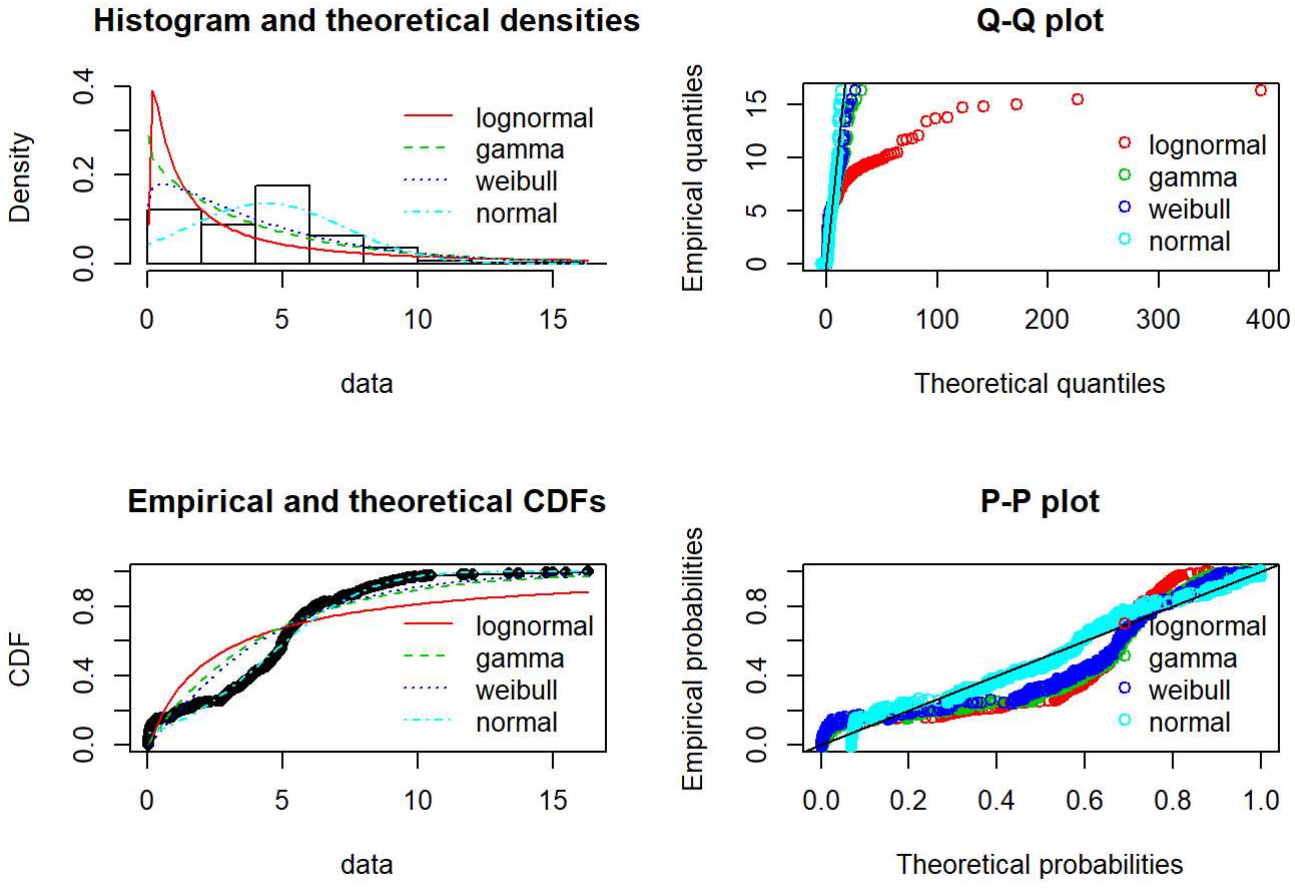
```
tmo3.fit.weibull <- fitdist(as.numeric(tmo3), "weibull")
tmo3.fit.weibull
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.104573  0.0400553
## scale 4.477054  0.1722414
```

```
tmo3.fit.normal <- fitdist(as.numeric(tmo3), "norm")
tmo3.fit.normal
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 4.360152 0.12073209
## sd   2.935056 0.08537043
```

```
par(mfrow=c(2,2))
plot.legend <- c("lognormal", "gamma", "weibull", "normal")
denscomp(list(tmo3.fit.lognormal, tmo3.fit.gamma, tmo3.fit.weibull, tmo3.fit.normal),
         legendtext = plot.legend)
qqcomp(list(tmo3.fit.lognormal, tmo3.fit.gamma, tmo3.fit.weibull, tmo3.fit.normal),
       legendtext = plot.legend)
cdfcomp(list(tmo3.fit.lognormal, tmo3.fit.gamma, tmo3.fit.weibull, tmo3.fit.normal),
        legendtext = plot.legend)
ppcomp(list(tmo3.fit.lognormal, tmo3.fit.gamma, tmo3.fit.weibull, tmo3.fit.normal),
       legendtext = plot.legend)
```




```
ks.test(tmo3, "pgamma", shape = tmo3.fit.gamma[["estimate"]][["shape"]],
        rate = tmo3.fit.gamma[["estimate"]][["rate"]])
```

```
## Warning in ks.test(tmo3, "pgamma", shape = tmo3.fit.gamma[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo3
## D = 0.21682, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(tmo3, "pweibull", shape = tmo3.fit.weibull[["estimate"]][["shape"]],
        scale = tmo3.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(tmo3, "pweibull", shape = tmo3.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo3
## D = 0.18634, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(tmo3, "pnorm", mean = tmo3.fit.normal[["estimate"]][["mean"]],
        sd = tmo3.fit.normal[["estimate"]][["sd"]])
```

```
## Warning in ks.test(tmo3, "pnorm", mean = tmo3.fit.normal[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  tmo3
## D = 0.072369, p-value = 0.004098
## alternative hypothesis: two-sided
```

The p-value for the KS Test of the data against the fitted normal distribution, using the estimated parameters from MLE, is the highest. This indicates that it is the closest fitted distribution to the data compared to the others. We determine the normal to be the best fitted disrtibution to describe the data. The Maximum Likelihood Estimation of the parameters when the data is fitted to a weibull distribution is a weibull distribution with mean = 4.36 and sd = 2.93

Problem 2e: Three Month Treasury Rate

When the data is subset based on recession periods, the data is best fit by a normal distribution in each case. But Maximum Likelihood Estimation found the parameters of these distributions to be different.

Problem 2f: Three Month Treasury Rate

```
ks.test(tmo, tmo2)
```

```
## Warning in ks.test(tmo, tmo2): p-value will be approximate in the presence
## of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  tmo and tmo2
## D = 0.36334, p-value = 1.534e-09
## alternative hypothesis: two-sided
```

```
ks.test(tmo, tmo3)
```

```
## Warning in ks.test(tmo, tmo3): p-value will be approximate in the presence
## of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  tmo and tmo3
## D = 0.055331, p-value = 0.2873
## alternative hypothesis: two-sided
```

```
ks.test(tmo2, tmo3)
```

```
## Warning in ks.test(tmo2, tmo3): p-value will be approximate in the presence
## of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  tmo2 and tmo3
## D = 0.41867, p-value = 2.567e-12
## alternative hypothesis: two-sided
```

The KS Tests comparing the three data samples indicate the the data containing all points and the data containing points generated during non-recession periods come from the same distribution.

Problem 2a: Japanese Central Bank Interest Rate

```
bank.rate <- read.csv("Japanese Bank Rate Fred.csv", header = TRUE)
date <- as.Date(bank.rate$DATE, format = "%Y-%m-%d")
bank.rate <- cbind(date, bank.rate)
bank.rate <- bank.rate[, -2]
names(bank.rate) <- paste(c("Date", "Bank.Rate"))
head(bank.rate, n = 3)
```

```
tail(bank.rate, n = 3)
```

```
nrow(bank.rate)
```

```
## [1] 705
```

```
summary(bank.rate$Bank.Rate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.100   0.300   3.500   3.336   5.840   9.000
```

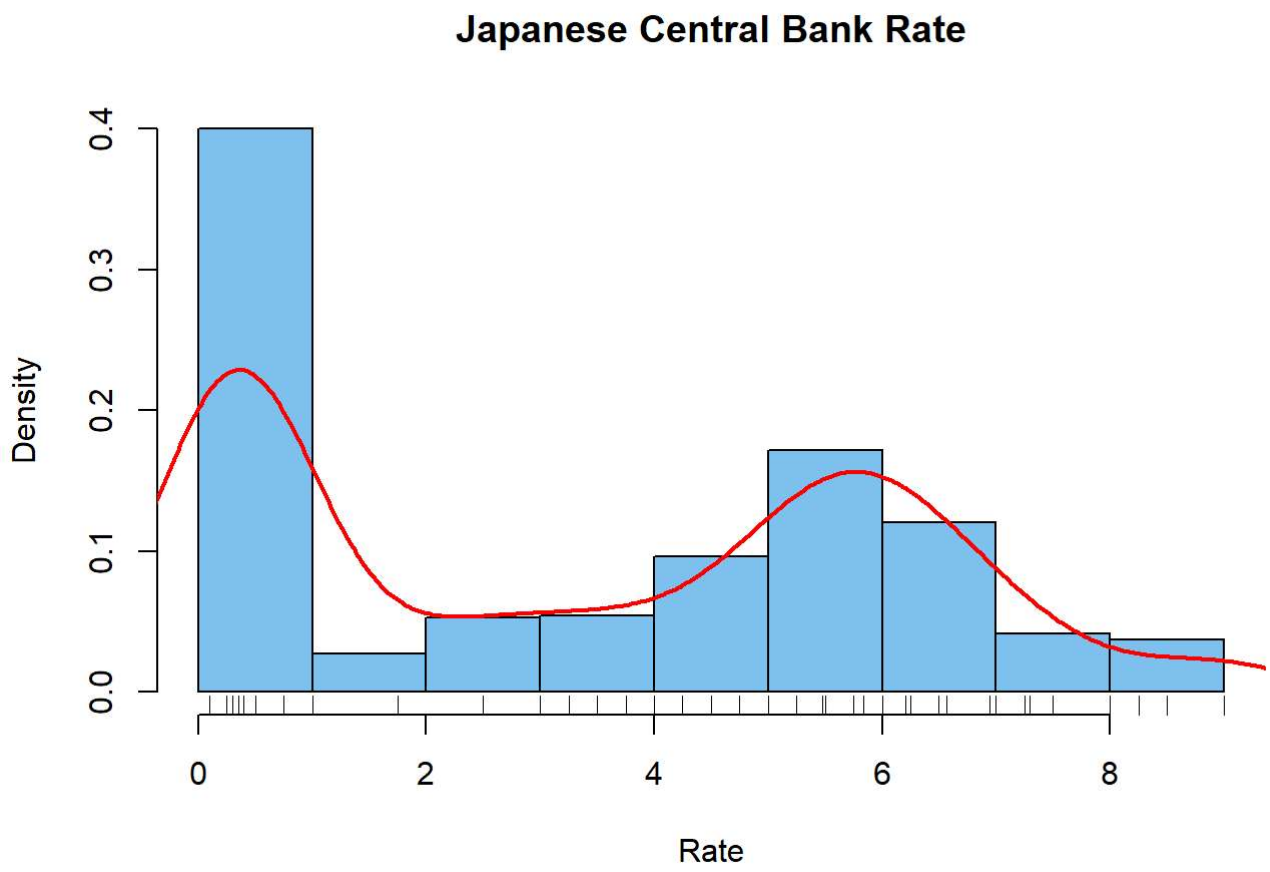
```
br <- bank.rate[, 2]
br <- as.vector(br)
length(br)
```

```
## [1] 705
```

```
summary(br)
```

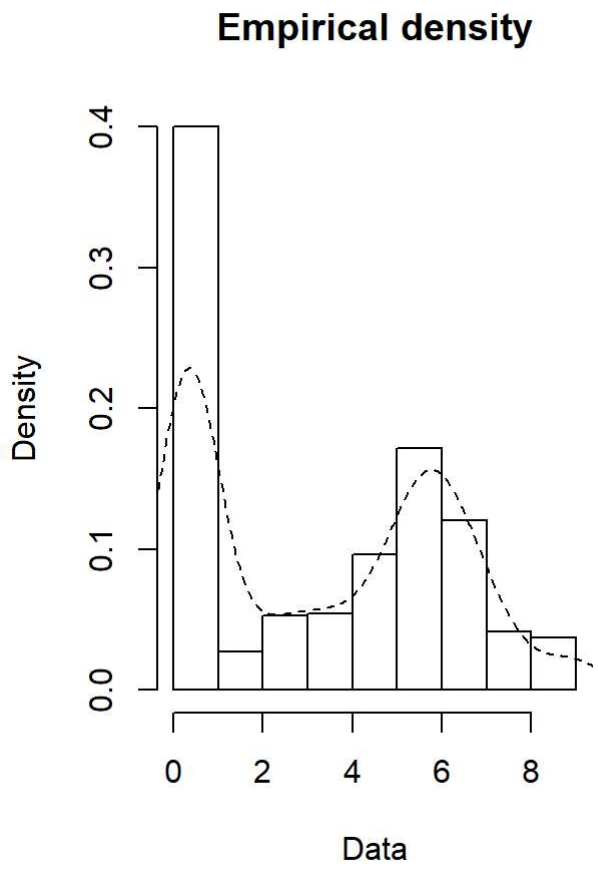
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.100   0.300   3.500   3.336   5.840   9.000
```

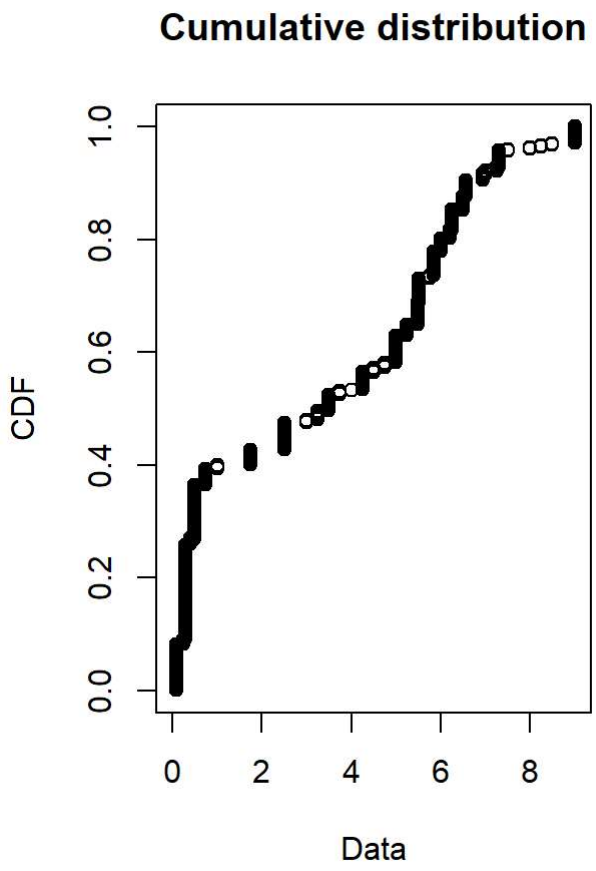
```
par(mfrow = c(1,1))
hist(br, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Japanese Central Bank Rate",
     xlab = "Rate")
lines(density(br), col = "red", lwd = 2)
rug(br)
```



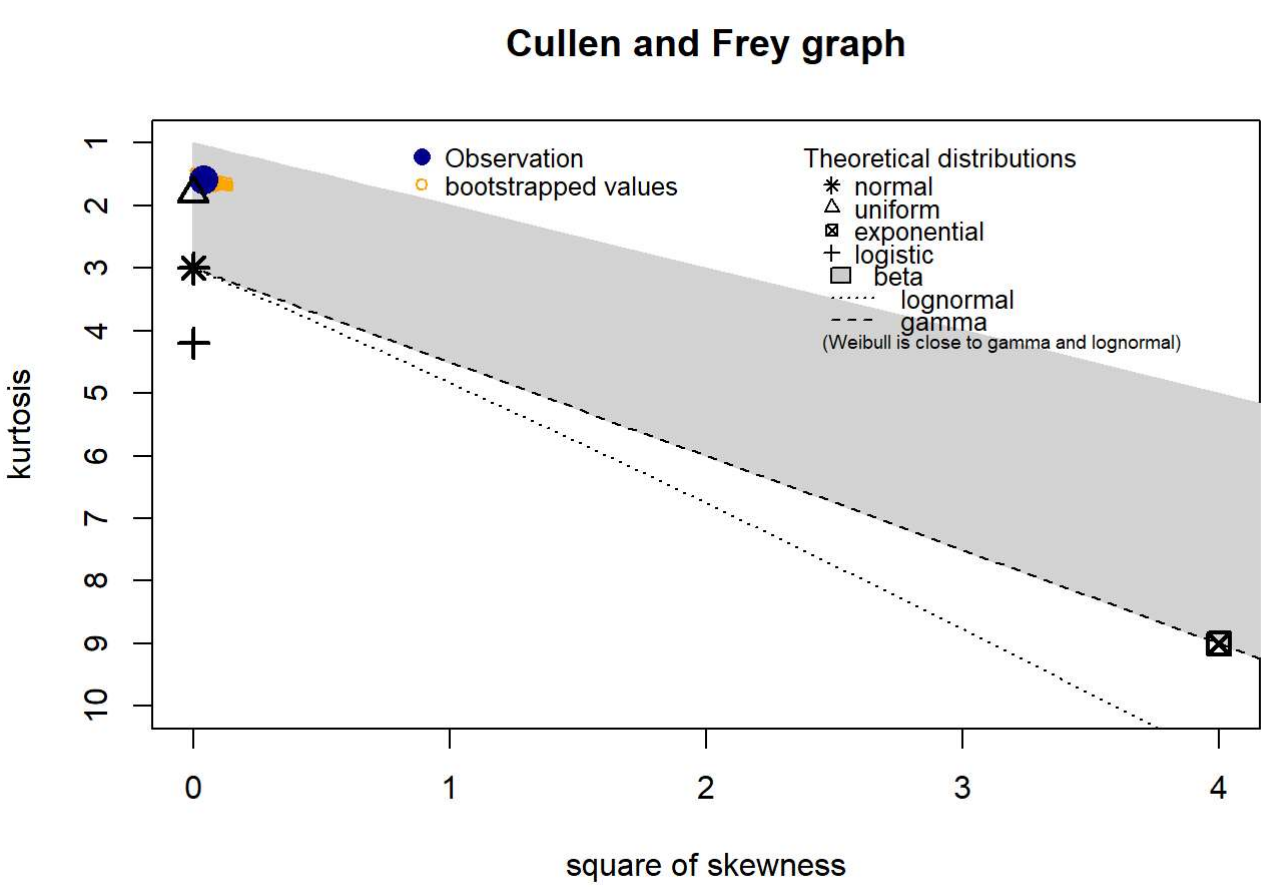
Problem 2b: Japanese Central Bank Interest Rate

```
plotdist(br, histo = TRUE, demp = TRUE)
```






```
descdist(br, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.1   max:  9
## median:  3.5
## mean:  3.336156
## estimated sd:  2.789298
## estimated skewness:  0.1982917
## estimated kurtosis:  1.592973
```

```
br.fit.uniform <- fitdist(as.numeric(br), "unif")
br.fit.uniform
```

```
## Fitting of the distribution ' unif ' by maximum likelihood
## Parameters:
##     estimate Std. Error
## min      0.1         NA
## max      9.0         NA
```

```
br.fit.lognormal <- fitdist(as.numeric(br), "lnorm")
br.fit.lognormal
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 0.4762574 0.05541962
## sdlog   1.4714926 0.03918751
```

```
br.fit.gamma <- fitdist(as.numeric(br), "gamma")
br.fit.gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.8126762 0.03740022
## rate  0.2436132 0.01514160
```

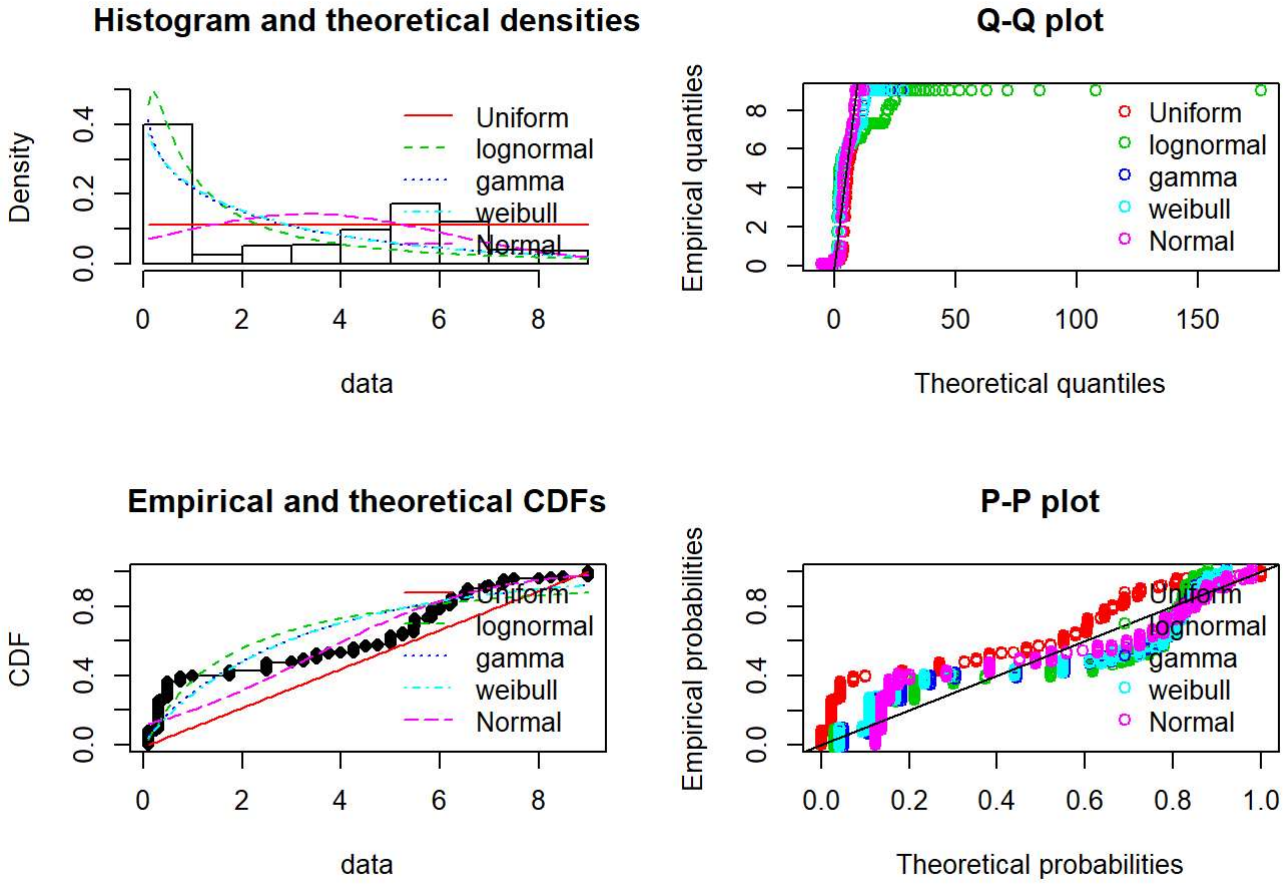
```
br.fit.weibull <- fitdist(as.numeric(br), "weibull")
br.fit.weibull
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.9061596 0.02903916
## scale 3.2024756 0.13977973
```

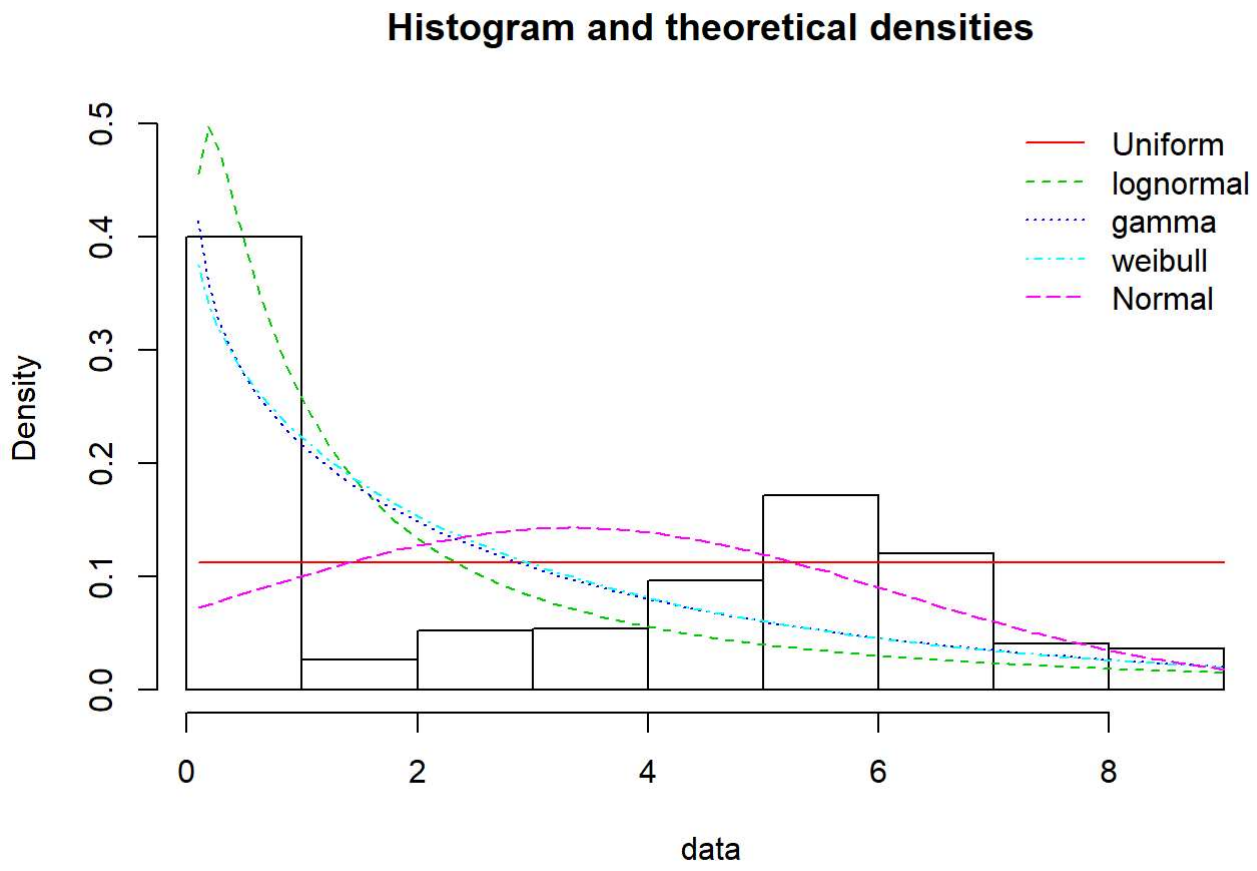
```
br.fit.norm <- fitdist(as.numeric(br), "norm")
br.fit.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 3.336156 0.10497652
## sd   2.787319 0.07422957
```

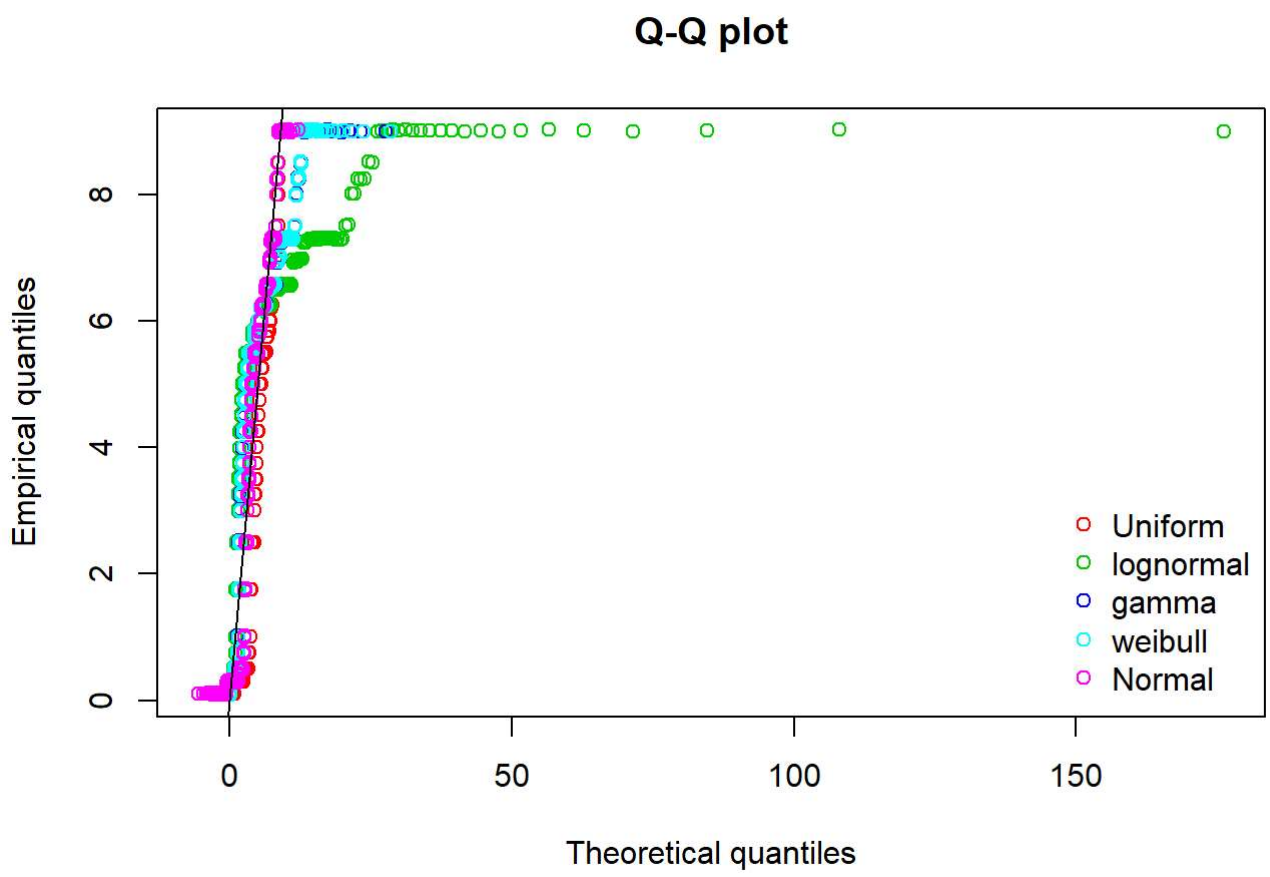
```
par(mfrow=c(2,2))
plot.legend <- c("Uniform", "lognormal", "gamma", "weibull", "Normal")
denscomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
              br.fit.norm), legendtext = plot.legend)
qqcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
            br.fit.norm), legendtext = plot.legend)
cdfcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
             br.fit.norm), legendtext = plot.legend)
ppcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
            br.fit.norm), legendtext = plot.legend)
```



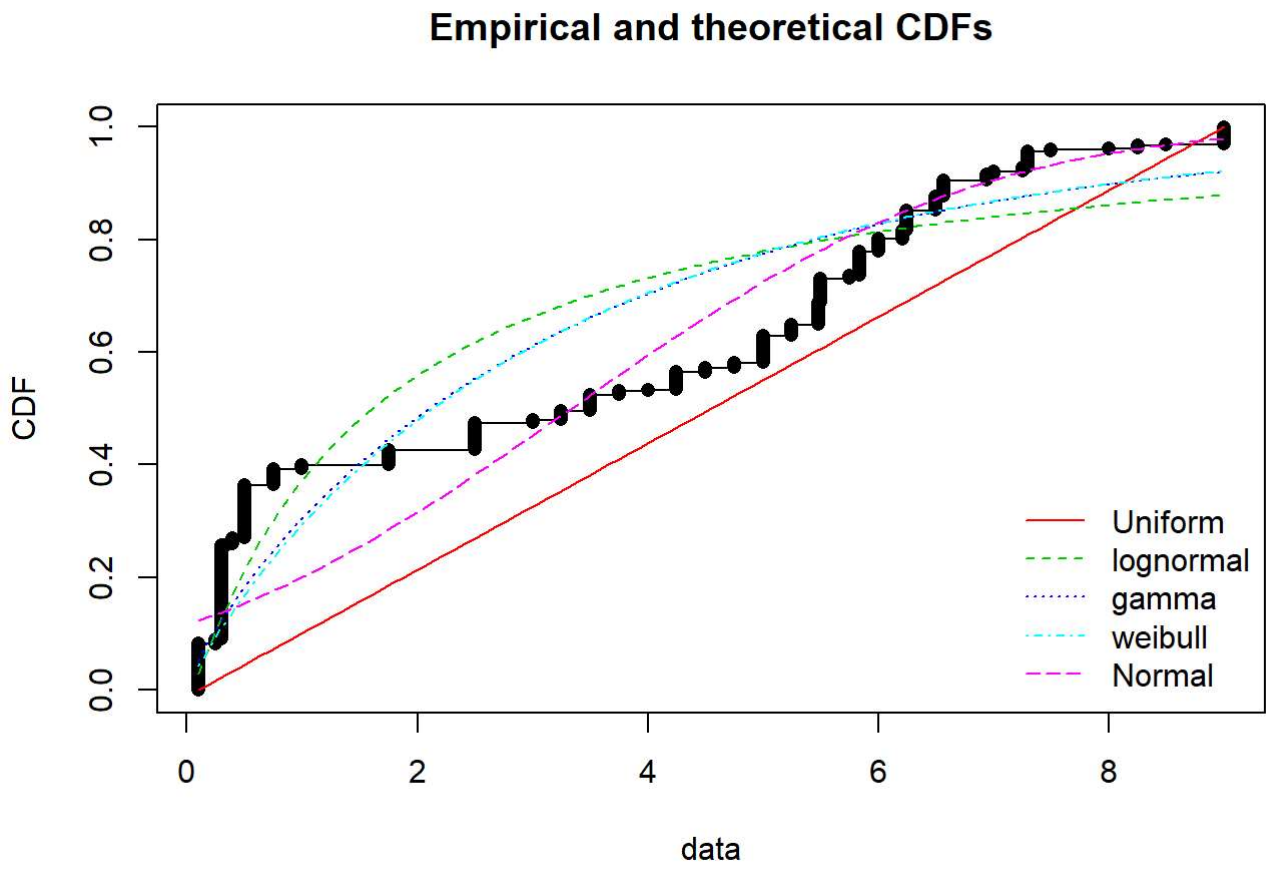

```
par(mfrow=c(1,1))
plot.legend <- c("Uniform", "lognormal", "gamma", "weibull", "Normal")
denscomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
              br.fit.norm), legendtext = plot.legend)
```



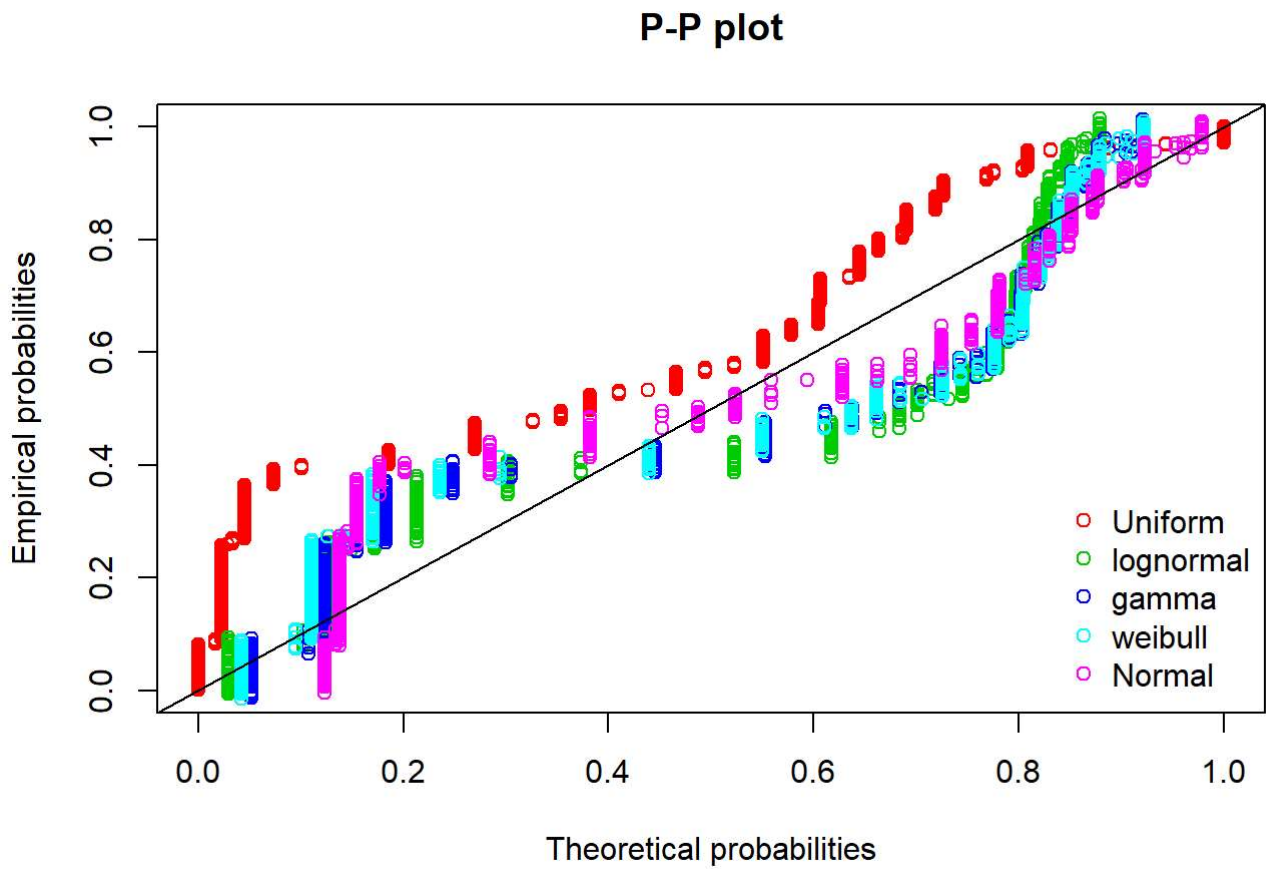

```
qqcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
            br.fit.norm), legendtext = plot.legend)
```




```
cdfcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
             br.fit.norm), legendtext = plot.legend)
```




```
ppcomp(list(br.fit.uniform, br.fit.lognormal, br.fit.gamma, br.fit.weibull,
            br.fit.norm), legendtext = plot.legend)
```




```
ks.test(br, "punif", min = br.fit.uniform[["estimate"]][["min"]],
        max = br.fit.uniform[["estimate"]][["max"]])
```

```
## Warning in ks.test(br, "punif", min = br.fit.uniform[["estimate"]]
## [["min"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br
## D = 0.31987, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br, "pnorm", mean = br.fit.norm[["estimate"]][["mean"]],
        sd = br.fit.norm[["estimate"]][["sd"]])
```

```
## Warning in ks.test(br, "pnorm", mean = br.fit.norm[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br
## D = 0.21616, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br, "pweibull", shape = br.fit.weibull[["estimate"]][["shape"]],
        scale = br.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(br, "pweibull", shape = br.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br
## D = 0.19494, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br, "pgamma", shape = br.fit.gamma[["estimate"]][["shape"]],
        rate = br.fit.gamma[["estimate"]][["rate"]])
```

```
## Warning in ks.test(br, "pgamma", shape = br.fit.gamma[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br
## D = 0.19308, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

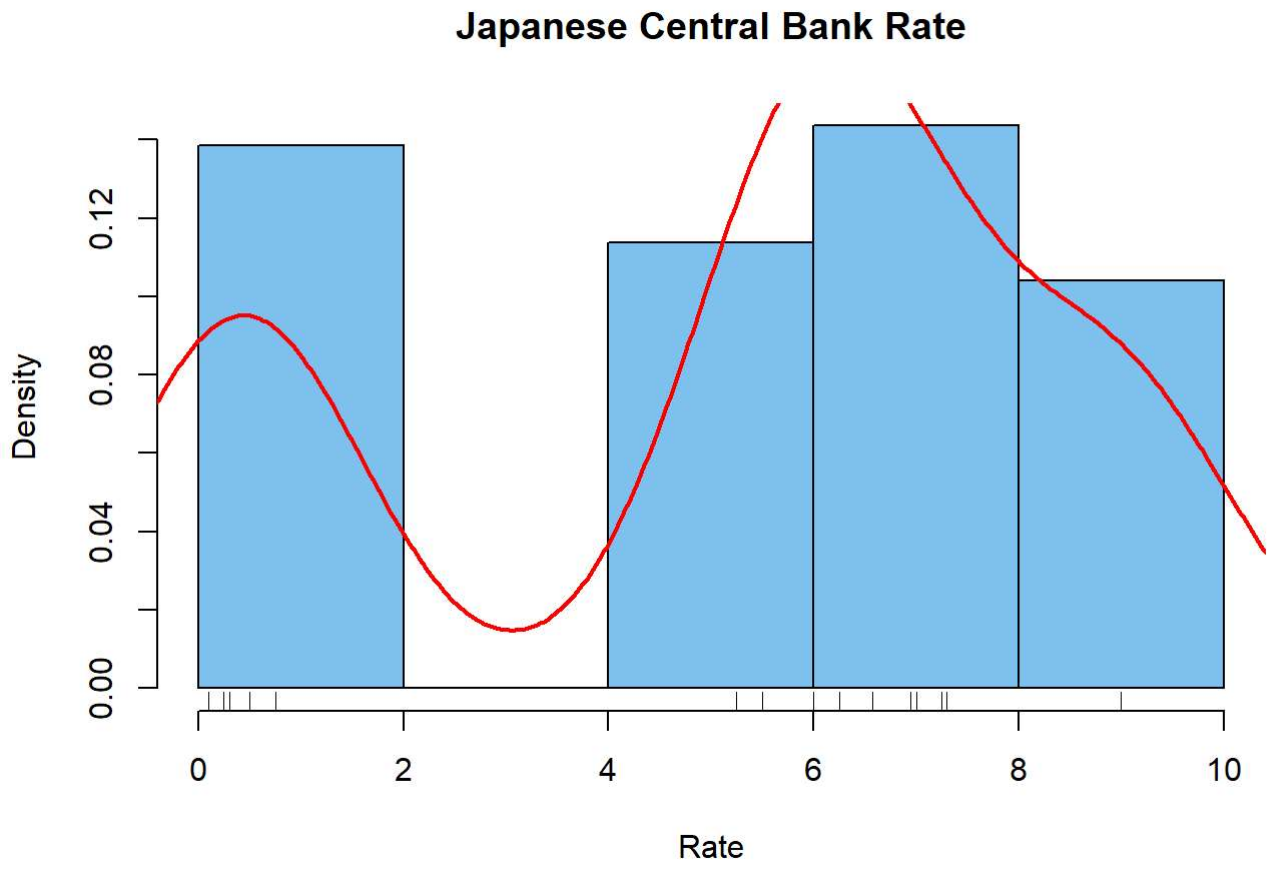
Based on the above, it appears that none of the fitted distributions very accuratly describe the data.

Problem 2c: Japanese Central Bank Interest Rate

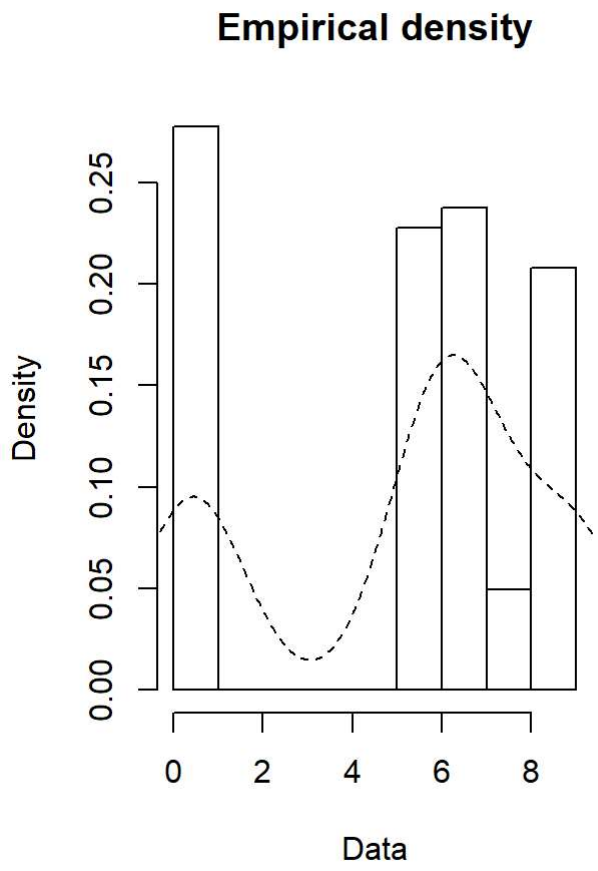
```
bank.rate2 <- bank.rate[bank.rate$Date %in% recession.dates$dates, ]
nrow(bank.rate2)
```

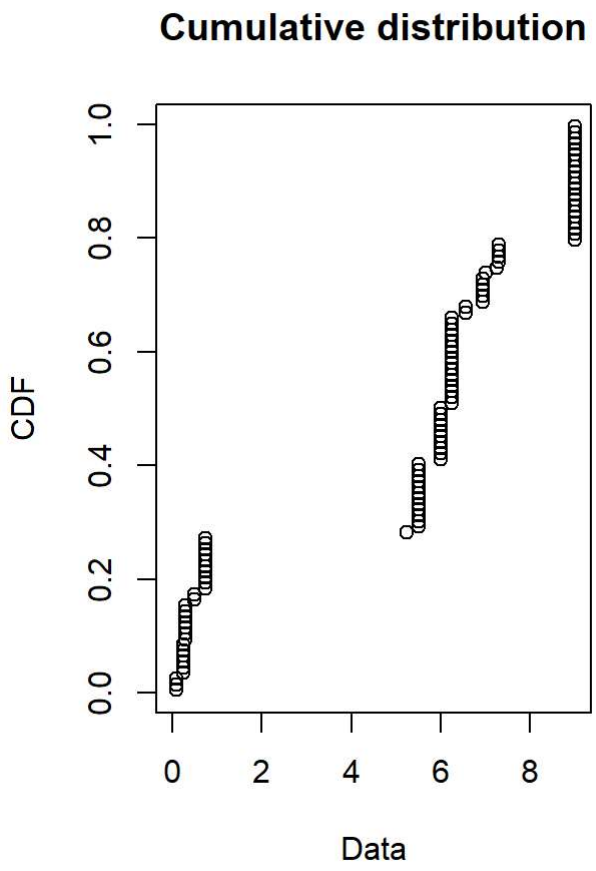
```
## [1] 101
```

```
br2 <- bank.rate2[, 2]
br2 <- as.vector(br2)
par(mfrow=c(1,1))
hist(br2, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Japanese Central Bank Rate",
     xlab = "Rate")
lines(density(br2), col = "red", lwd = 2)
rug(br2)
```

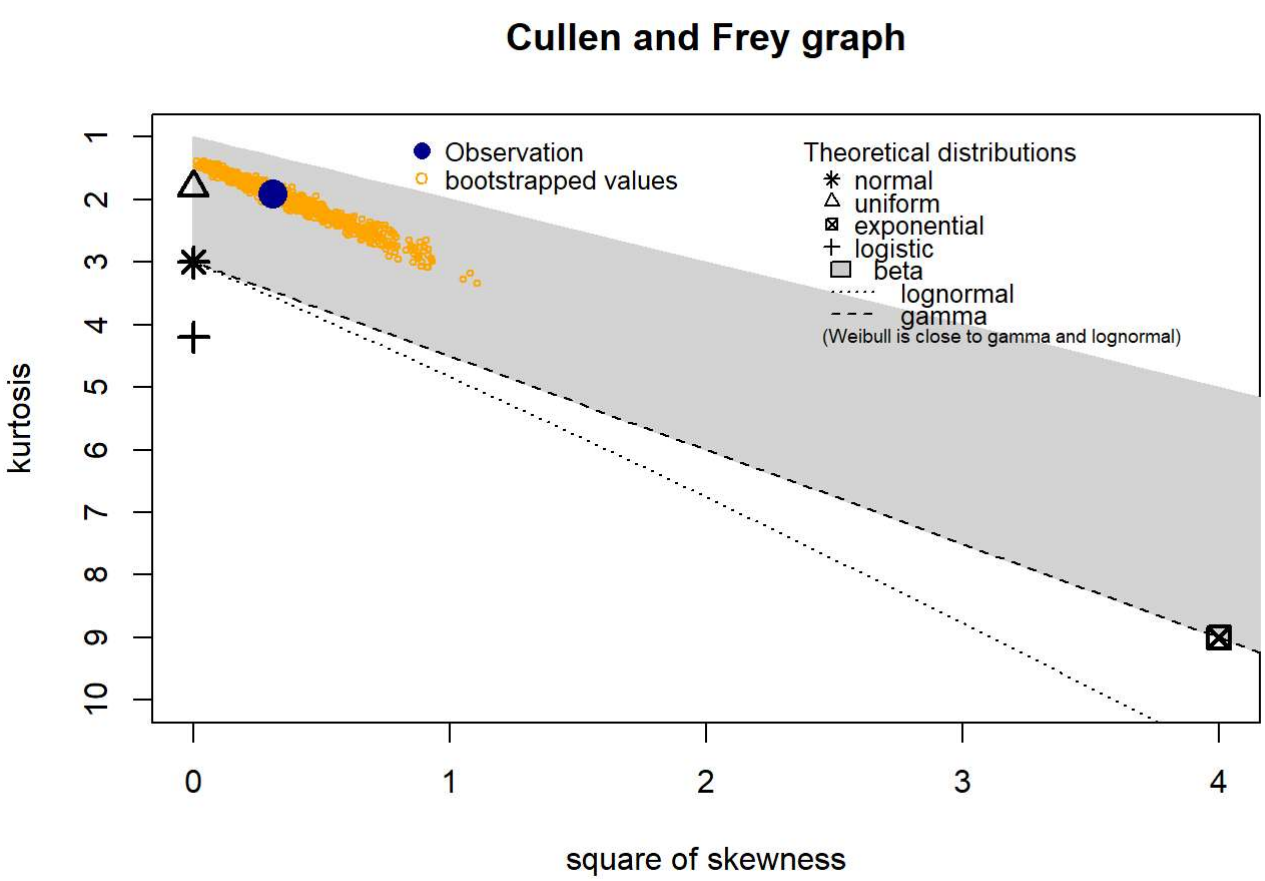



```
plotdist(br2, histo = TRUE, demp = TRUE)
```






```
descdist(br2, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.1   max:  9
## median:  6
## mean:  5.187525
## estimated sd:  3.174336
## estimated skewness:  -0.5553343
## estimated kurtosis:  1.898839
```

```
br2.fit.uniform <- fitdist(as.numeric(br2), "unif")
br2.fit.uniform
```

```
## Fitting of the distribution ' unif ' by maximum likelihood
## Parameters:
##     estimate Std. Error
## min      0.1         NA
## max      9.0         NA
```

```
br2.fit.lognormal <- fitdist(as.numeric(br2), "lnorm")
br2.fit.lognormal
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##         estimate Std. Error
## meanlog 1.117914 0.13552610
## sdlog   1.362020 0.09583119
```

```
br2.fit.gamma <- fitdist(as.numeric(br2), "gamma")
br2.fit.gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 1.0828185 0.13509811
## rate  0.2087511 0.03281345
```

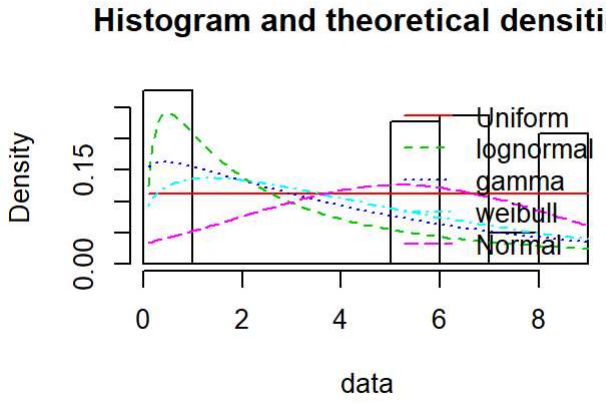
```
br2.fit.weibull <- fitdist(as.numeric(br2), "weibull")
br2.fit.weibull
```

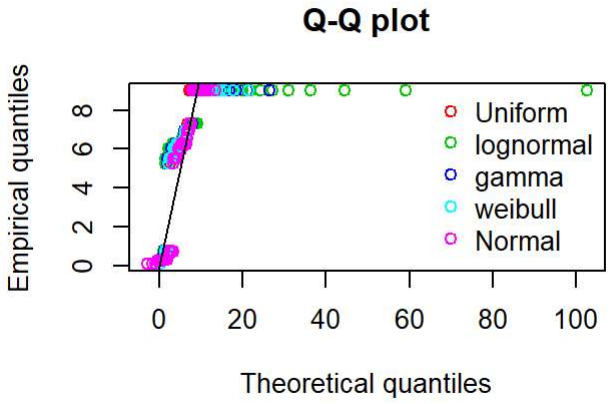
```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 1.218881  0.1111551
## scale 5.450272  0.4609513
```

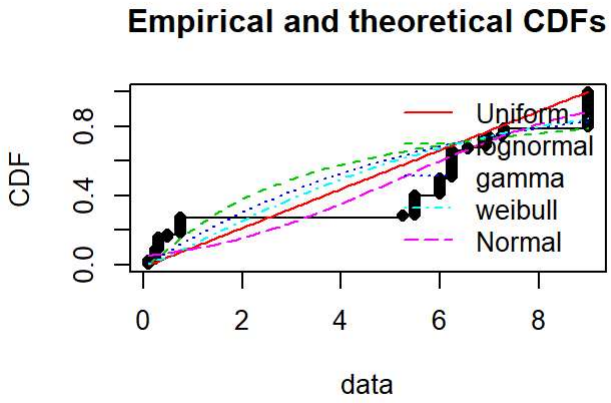
```
br2.fit.norm <- fitdist(as.numeric(br2), "norm")
br2.fit.norm
```

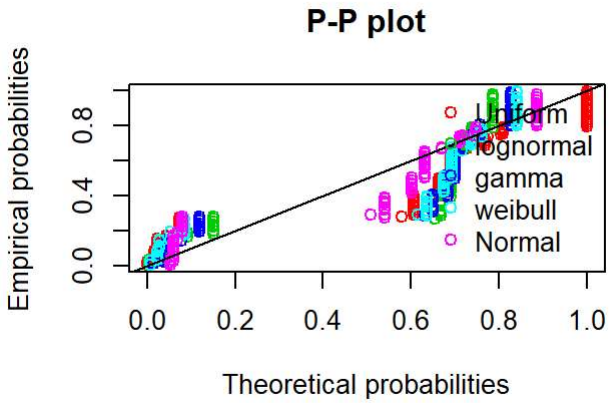
```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 5.187525  0.3142907
## sd   3.158583  0.2222370
```

```
par(mfrow=c(2,2))
plot.legend <- c("Uniform", "lognormal", "gamma", "weibull", "Normal")
denscomp(list(br2.fit.uniform, br2.fit.lognormal, br2.fit.gamma, br2.fit.weibull,
              br2.fit.norm), legendtext = plot.legend)
qqcomp(list(br2.fit.uniform, br2.fit.lognormal, br2.fit.gamma, br2.fit.weibull,
            br2.fit.norm), legendtext = plot.legend)
cdfcomp(list(br2.fit.uniform, br2.fit.lognormal, br2.fit.gamma, br2.fit.weibull,
             br2.fit.norm), legendtext = plot.legend)
ppcomp(list(br2.fit.uniform, br2.fit.lognormal, br2.fit.gamma, br2.fit.weibull,
            br2.fit.norm), legendtext = plot.legend)
```










```
ks.test(br2, "punif", min = br2.fit.uniform[["estimate"]][["min"]],
        max = br2.fit.uniform[["estimate"]][["max"]])
```

```
## Warning in ks.test(br2, "punif", min = br2.fit.uniform[["estimate"]]
## [["min"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br2
## D = 0.31961, p-value = 2.185e-09
## alternative hypothesis: two-sided
```

```
ks.test(br2, "pnorm", mean = br2.fit.norm[["estimate"]][["mean"]],
        sd = br2.fit.norm[["estimate"]][["sd"]])
```

```
## Warning in ks.test(br2, "pnorm", mean = br2.fit.norm[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br2
## D = 0.25227, p-value = 5.222e-06
## alternative hypothesis: two-sided
```

```
ks.test(br2, "pweibull", shape = br2.fit.weibull[["estimate"]][["shape"]],
        scale = br2.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(br2, "pweibull", shape = br2.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br2
## D = 0.34906, p-value = 4.091e-11
## alternative hypothesis: two-sided
```

```
ks.test(br2, "pgamma", shape = br2.fit.gamma[["estimate"]][["shape"]],
        rate = br2.fit.gamma[["estimate"]][["rate"]])
```

```
## Warning in ks.test(br2, "pgamma", shape = br2.fit.gamma[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br2
## D = 0.36254, p-value = 5.9e-12
## alternative hypothesis: two-sided
```

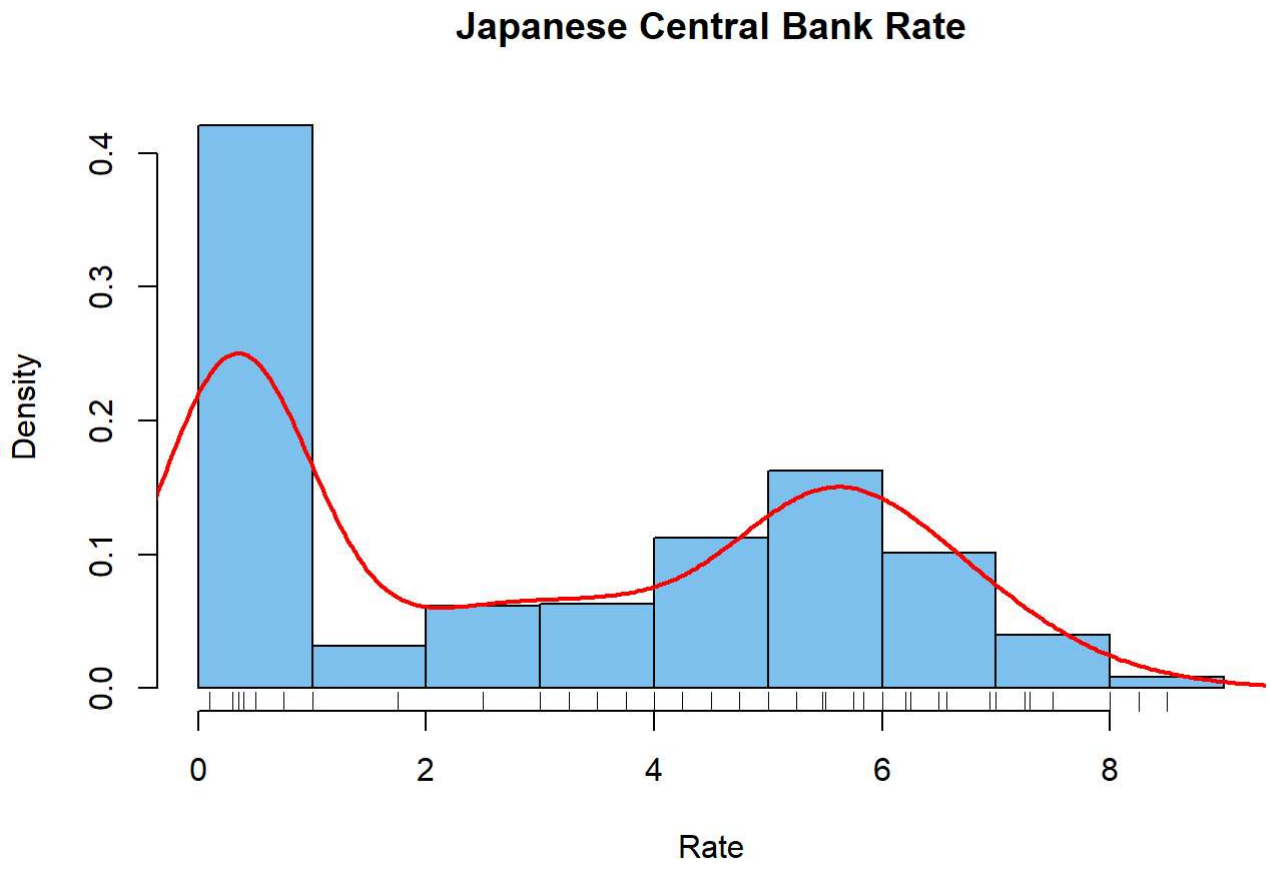
Based on the above, it appears that none of the fitted distributions very accuratly describe the data, which includes only points generated during recession periods. However, the uniform distribution appears to describe the data the best. The Maximum Likelihood Estimation of the parameters when the data is fitted to a normal distribution is a uniform with min = 0.1 and max = 9.0

Problem 2d: Japanese Cental Bank Interst Rate

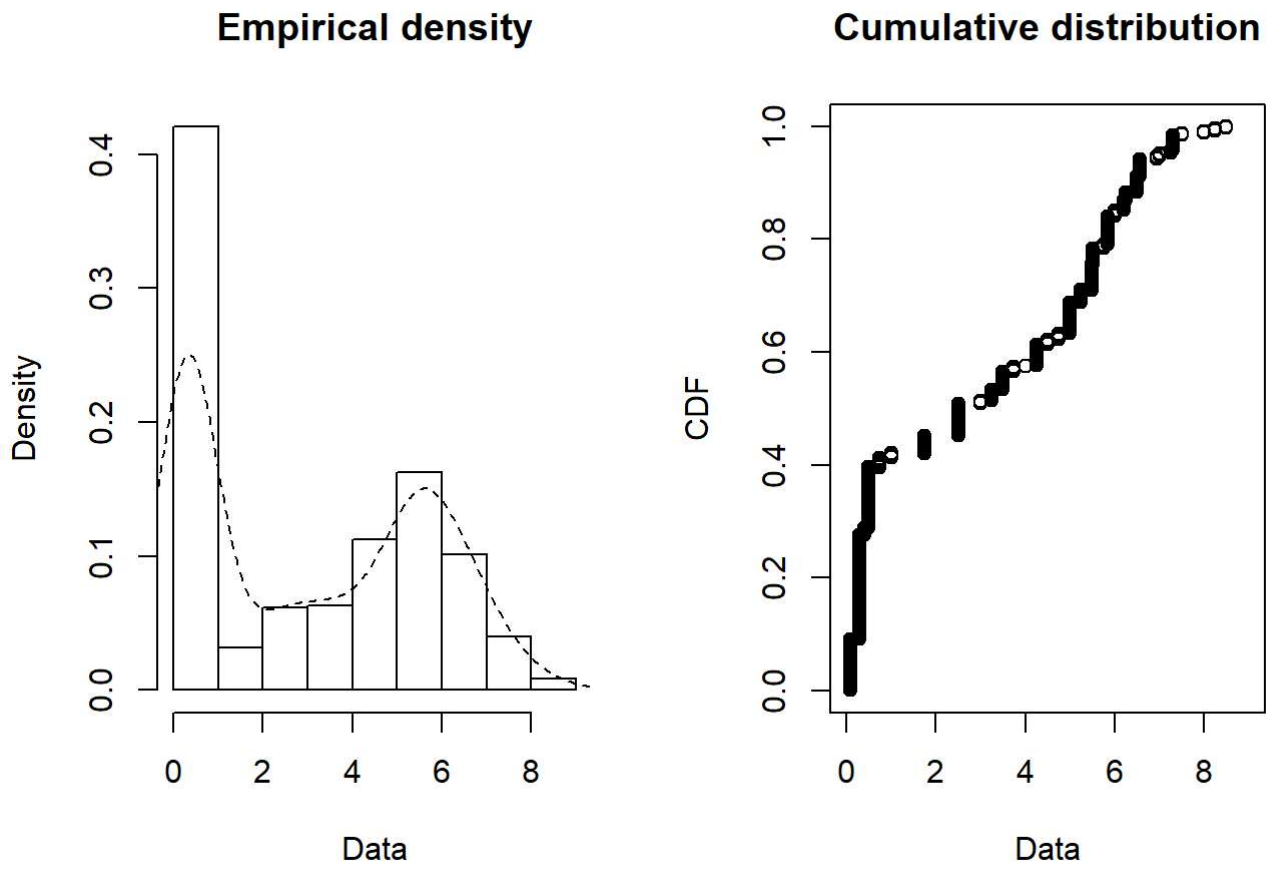
```
bank.rate3 <- bank.rate[!(bank.rate$Date %in% recession.dates$dates), ]
nrow(bank.rate3)
```

```
## [1] 604
```

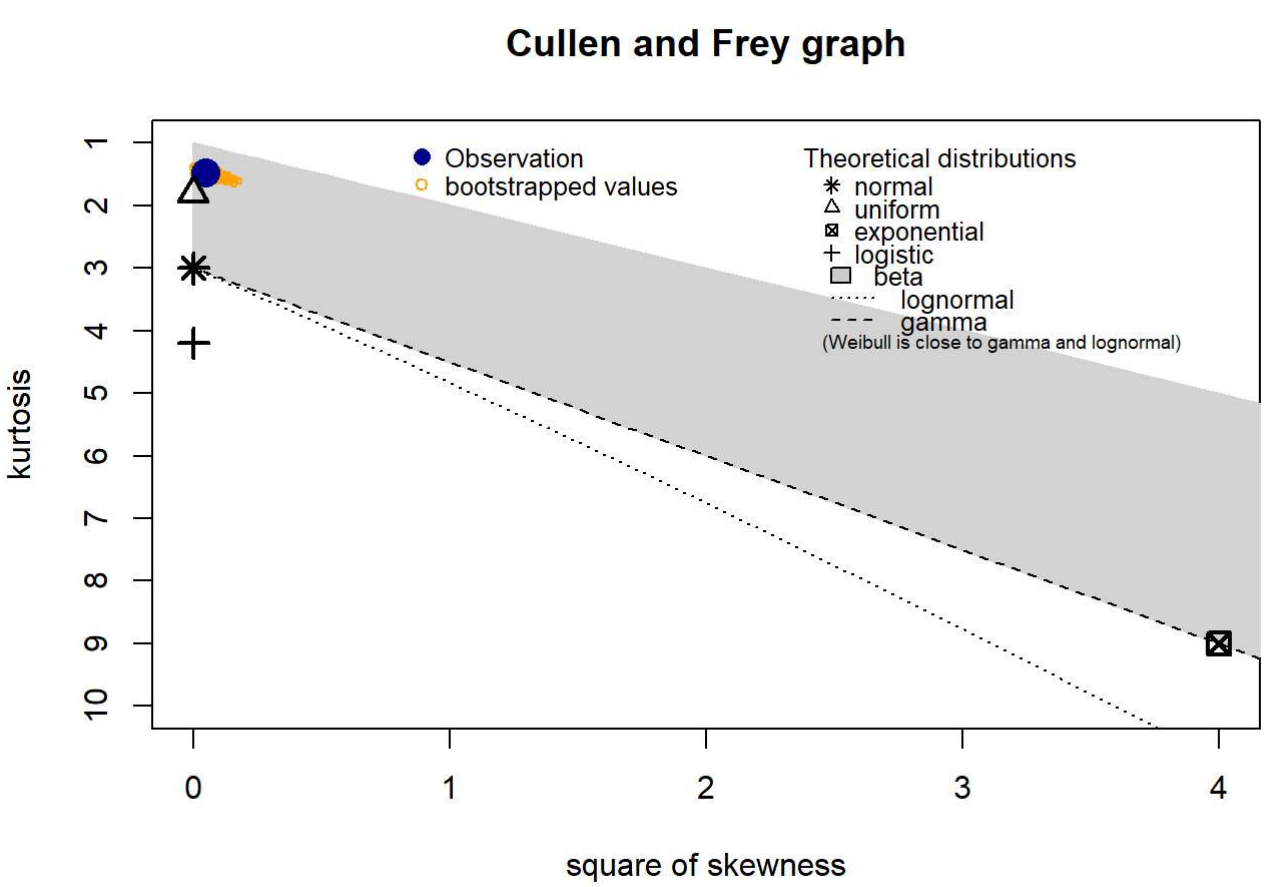
```
br3 <- bank.rate3[, 2]
br3 <- as.vector(br3)
par(mfrow=c(1,1))
hist(br3, breaks = "FD", prob = TRUE, col = "skyblue2", main = "Japanese Central Bank Rate",
     xlab = "Rate")
lines(density(br3), col = "red", lwd = 2)
rug(br3)
```




```
plotdist(br3, histo = TRUE, demp = TRUE)
```




```
descdist(br3, boot = 1000)
```




```
## summary statistics
## ------
## min:  0.1   max:  8.5
## median:  2.5
## mean:  3.026573
## estimated sd:  2.596571
## estimated skewness:  0.2203377
## estimated kurtosis:  1.4708
```

```
br3.fit.uniform <- fitdist(as.numeric(br3), "unif")
br3.fit.uniform
```

```
## Fitting of the distribution ' unif ' by maximum likelihood
## Parameters:
##     estimate Std. Error
## min      0.1         NA
## max      8.5         NA
```

```
br3.fit.lognormal <- fitdist(as.numeric(br3), "lnorm")
br3.fit.lognormal
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 0.3689606 0.05947897
## sdlog   1.4617796 0.04205789
```

```
br3.fit.gamma <- fitdist(as.numeric(br3), "gamma")
br3.fit.gamma
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.8033278 0.03989882
## rate  0.2654368 0.01786055
```

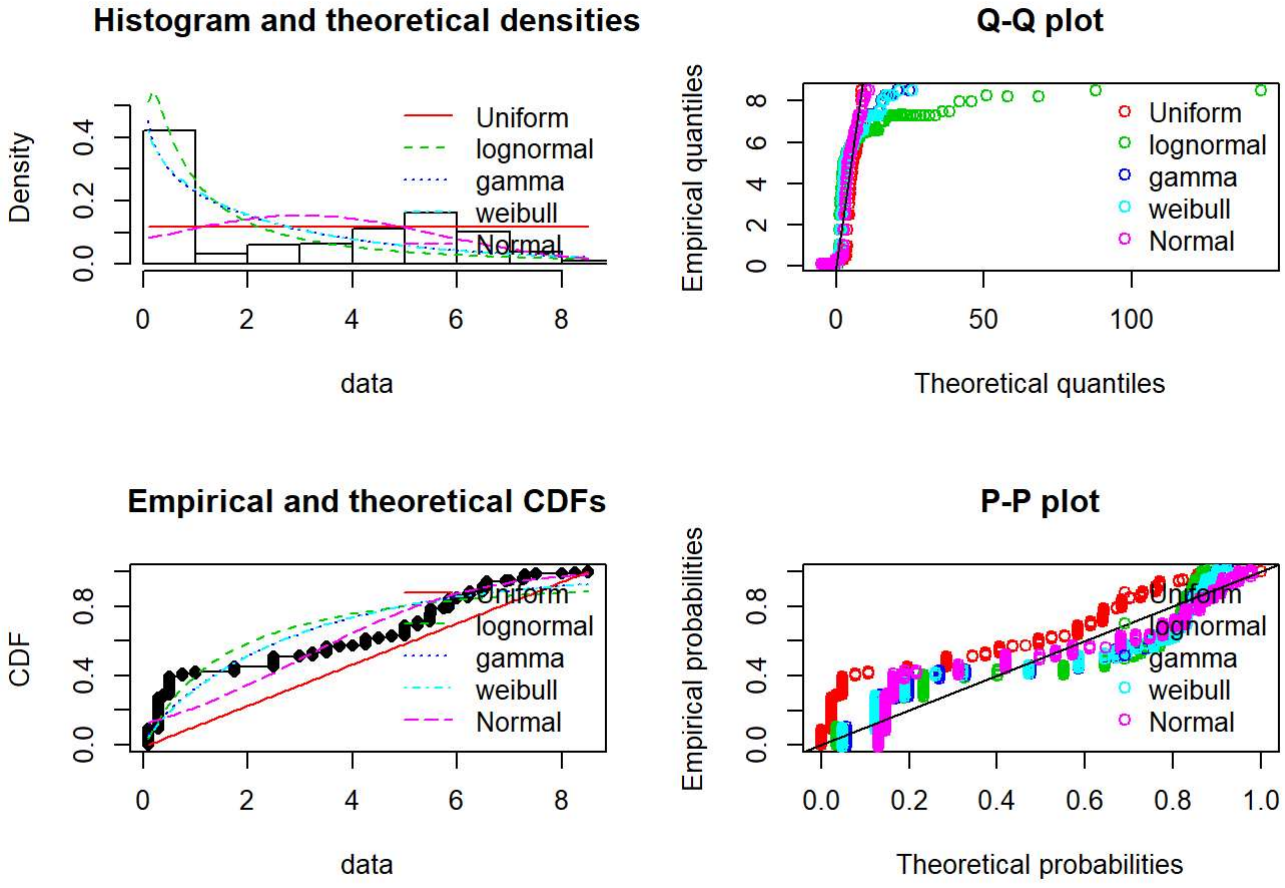
```
br3.fit.weibull <- fitdist(as.numeric(br3), "weibull")
br3.fit.weibull
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##       estimate Std. Error
## shape 0.892656 0.03066804
## scale 2.882604 0.13816540
```

```
br3.fit.norm <- fitdist(as.numeric(br3), "norm")
br3.fit.norm
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters:
##      estimate Std. Error
## mean 3.026573 0.10556549
## sd   2.594421 0.07464603
```

```
par(mfrow=c(2,2))
plot.legend <- c("Uniform", "lognormal", "gamma", "weibull", "Normal")
denscomp(list(br3.fit.uniform, br3.fit.lognormal, br3.fit.gamma, br3.fit.weibull,
              br3.fit.norm), legendtext = plot.legend)
qqcomp(list(br3.fit.uniform, br3.fit.lognormal, br3.fit.gamma, br3.fit.weibull,
            br3.fit.norm), legendtext = plot.legend)
cdfcomp(list(br3.fit.uniform, br3.fit.lognormal, br3.fit.gamma, br3.fit.weibull,
             br3.fit.norm), legendtext = plot.legend)
ppcomp(list(br3.fit.uniform, br3.fit.lognormal, br3.fit.gamma, br3.fit.weibull,
            br3.fit.norm), legendtext = plot.legend)
```




```
ks.test(br3, "punif", min = br3.fit.uniform[["estimate"]][["min"]],
        max = br3.fit.uniform[["estimate"]][["max"]])
```

```
## Warning in ks.test(br3, "punif", min = br3.fit.uniform[["estimate"]]
## [["min"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br3
## D = 0.34808, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br3, "pnorm", mean = br3.fit.norm[["estimate"]][["mean"]],
        sd = br3.fit.norm[["estimate"]][["sd"]])
```

```
## Warning in ks.test(br3, "pnorm", mean = br3.fit.norm[["estimate"]]
## [["mean"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br3
## D = 0.23063, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br3, "pweibull", shape = br3.fit.weibull[["estimate"]][["shape"]],
        scale = br3.fit.weibull[["estimate"]][["scale"]])
```

```
## Warning in ks.test(br3, "pweibull", shape = br3.fit.weibull[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br3
## D = 0.20681, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(br3, "pgamma", shape = br3.fit.gamma[["estimate"]][["shape"]],
        rate = br3.fit.gamma[["estimate"]][["rate"]])
```

```
## Warning in ks.test(br3, "pgamma", shape = br3.fit.gamma[["estimate"]]
## [["shape"]], : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  br3
## D = 0.19592, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Based on the above, it appears that none of the fitted distributions very accuratly describe the data, which includes only points generated during non-recession periods. The p-values for the KS Tests on the data against each of the fitted distributions is identical.

Problem 2e: Japanese Central Bank Interest Rate

It appears that none of the fitted distributions using MLE estimates for the parameters accuratly describes the three subsetted data sets.

Problem 2f: Japanese Central Bank Interest Rate

```
ks.test(br, br2)
```

```
## Warning in ks.test(br, br2): p-value will be approximate in the presence of
## ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  br and br2
## D = 0.40081, p-value = 9.408e-13
## alternative hypothesis: two-sided
```

```
ks.test(br, br3)
```

```
## Warning in ks.test(br, br3): p-value will be approximate in the presence of
## ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  br and br3
## D = 0.067024, p-value = 0.1076
## alternative hypothesis: two-sided
```

```
ks.test(br2, br3)
```

```
## Warning in ks.test(br2, br3): p-value will be approximate in the presence
## of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  br2 and br3
## D = 0.46784, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

The KS Tests indicate that the data containing all points and the data containing only points generated during the non-recession periods come from a similar distribution. However, the data containing only points generated during a recession does not follow the same distribution as the other two data sets.

Question 3 Ordinary Least Squares: A Lesson on CAPM and Parameter Stability/Robustness

The capital asset pricing model (CAPM) is an important model in the field of finance. It explains variations in the rate of return on a security as a function of the rate of return on a portfolio consisting of all publicly traded stocks, which is called the market portfolio. Generally the rate of return on any investment is measured relative to its opportunity cost, which is the return on a risk free asset. The resulting difference is called the risk premium, since it is the reward or punishment for making a risky investment. The CAPM says that the risk premium on security j is proportional to the risk premium on the market portfolio. That is,

$$r_j - r_f = \beta_j(r_m - r_f)$$

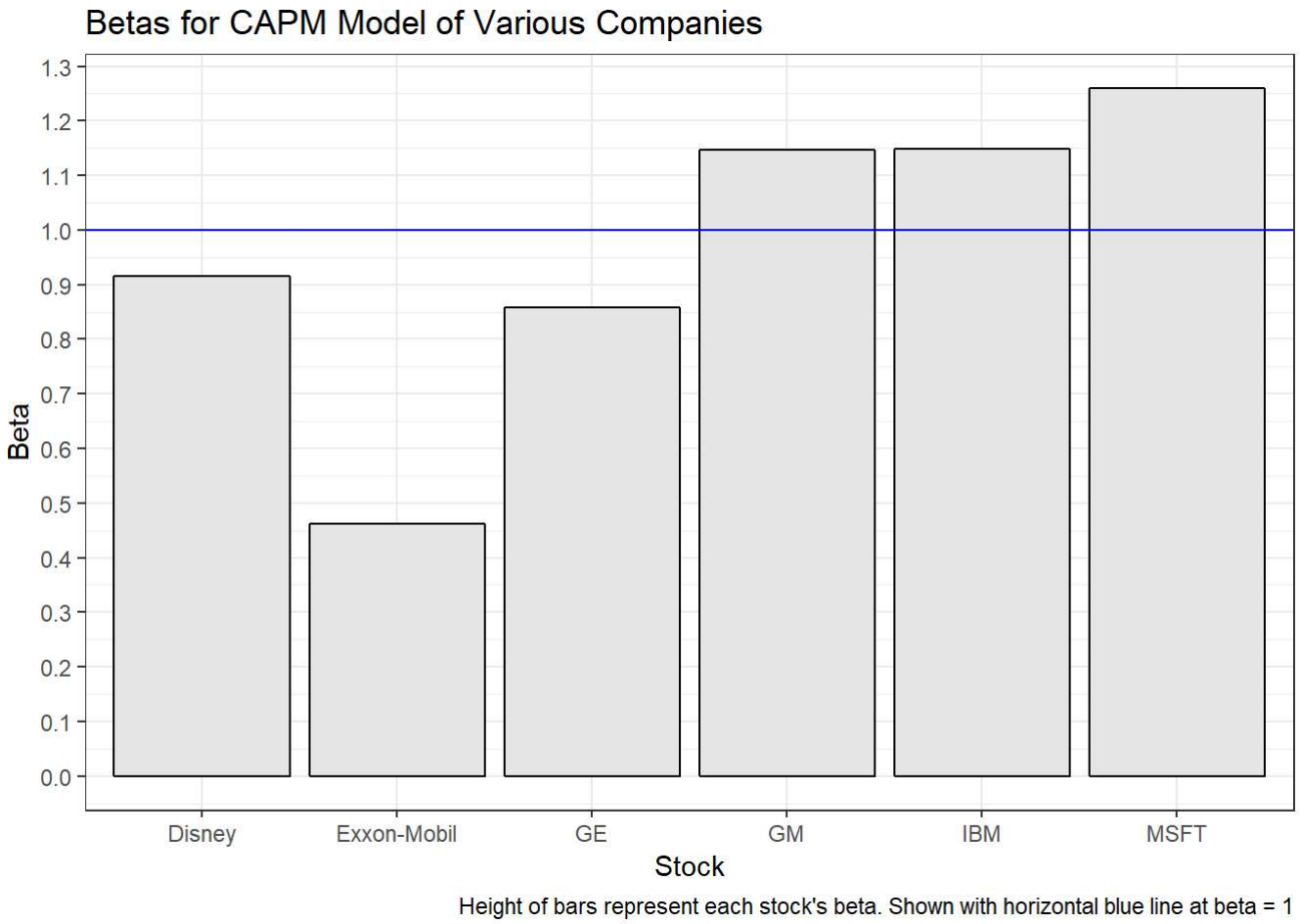
where $\mathbf{r_j}$ and $\mathbf{r_f}$ are the returns to security $\mathbf{j}$ and the risk-free rate, respectively, $\mathbf{r_m}$ is the return on the market portfolio, and $\beta_\mathbf{j}$ is the $\mathbf{j}$th security's "beta" value. A stocks beta is important to investors since it reveals the stock's volatility. It measures the sensitiviy of security $\mathbf{j}$'s return on the variation in the whole stock market. As such, values of beta less than 1 indicate that the stock is "defensive" since its variation is less than the market's. A beta greater than 1 indicates an "aggressive" stock. Investors usually want an estimate of a stock's beta before purchasing it. The CAPM model shown above is the "economic" model in this case. The "econometric" model is obtained by including an intercept in the model (even though theory says it should be zero) and an error term,

$$r_j - r_f = \alpha_j + \beta_j(r_m - r_f) + e$$

(a) The data file capm4.dat contains monthly returns of six firms (Microsoft, GE, GM, IBM, Disney, and Mobil-Exxon), the rate of return on the market portfolio (MKT), and the rate of return on the risk free asset (RISKFREE). The 132 observations cover January 1998 to December 2008. Estimate the CAPM model for each firm, and comment on their estimated beta values. Which firm appears most aggressive? Which firrm appears most defensive?

```
#read in and prep the data
market = read.csv("capm4.csv", stringsAsFactors = F)
market$mktfree = market$mkt - market$riskfree
market$disfree = market$dis-market$riskfree
market$gefree = market$ge-market$riskfree
market$gmfree = market$gm-market$riskfree
market$ibmfree = market$ibm-market$riskfree
market$xomfree = market$xom-market$riskfree
market$msftfree = market$msft-market$riskfree

#estimate the model for each company
micsoft = lm(msftfree ~ mktfree, data = market)
GE = lm(gefree ~ mktfree, data = market)
GM = lm(gmfree ~ mktfree, data = market)
IBM = lm(ibmfree ~ mktfree, data = market)
Disney = lm(disfree ~ mktfree, data = market)
ExMobil = lm(xomfree ~ mktfree, data = market)

#store betas in df
beta = data.frame(
  stock = c("MSFT", "GE", "GM", "IBM", "Disney", "Exxon-Mobil"),
  beta = c(micsoft[[1]][2], GE[[1]][2], GM[[1]][2], IBM[[1]][2], Disney[[1]][2], ExMobil[[1]][2
]),
  stde = c(summary(micsoft)$coefficients[2,2], summary(GE)$coefficients[2,2], summary(GM)$coeffi
cients[2,2], summary(IBM)$coefficients[2,2], summary(Disney)$coefficients[2,2], summary(ExMobil)
$coefficients[2,2]),
  confup = NA,
  conflow = NA
  )

alpha = data.frame(
  stock = c("MSFT", "GE", "GM", "IBM", "Disney", "Exxon-Mobil"),
  alpha = c(micsoft[[1]][1], GE[[1]][1], GM[[1]][1], IBM[[1]][1], Disney[[1]][1], ExMobil[[1]][1
]),
  stde = c(summary(micsoft)$coefficients[1,2], summary(GE)$coefficients[1,2], summary(GM)$coeffi
cients[1,2], summary(IBM)$coefficients[1,2], summary(Disney)$coefficients[1,2], summary(ExMobil)
$coefficients[1,2]),
  confup = NA,
  conflow = NA
)

for(i in 1:nrow(beta)){
  beta[i, "confup"] = beta[i, "beta"] + abs(qt(0.025,130))*beta[i, "stde"]
  beta[i, "conflow"] = beta[i, "beta"] - abs(qt(0.025,130))*beta[i, "stde"]
  alpha[i, "confup"] = alpha[i, "alpha"] + abs(qt(0.025,130))*alpha[i, "stde"]
  alpha[i, "conflow"] = alpha[i, "alpha"] - abs(qt(0.025,130))*alpha[i, "stde"]
}

#show betas graphically
ggplot(beta)+
  geom_col(aes(x = stock, y = beta), color = "black", fill = "grey90")+
  geom_hline(yintercept = 1, color = "blue")+
  theme_bw()+
```

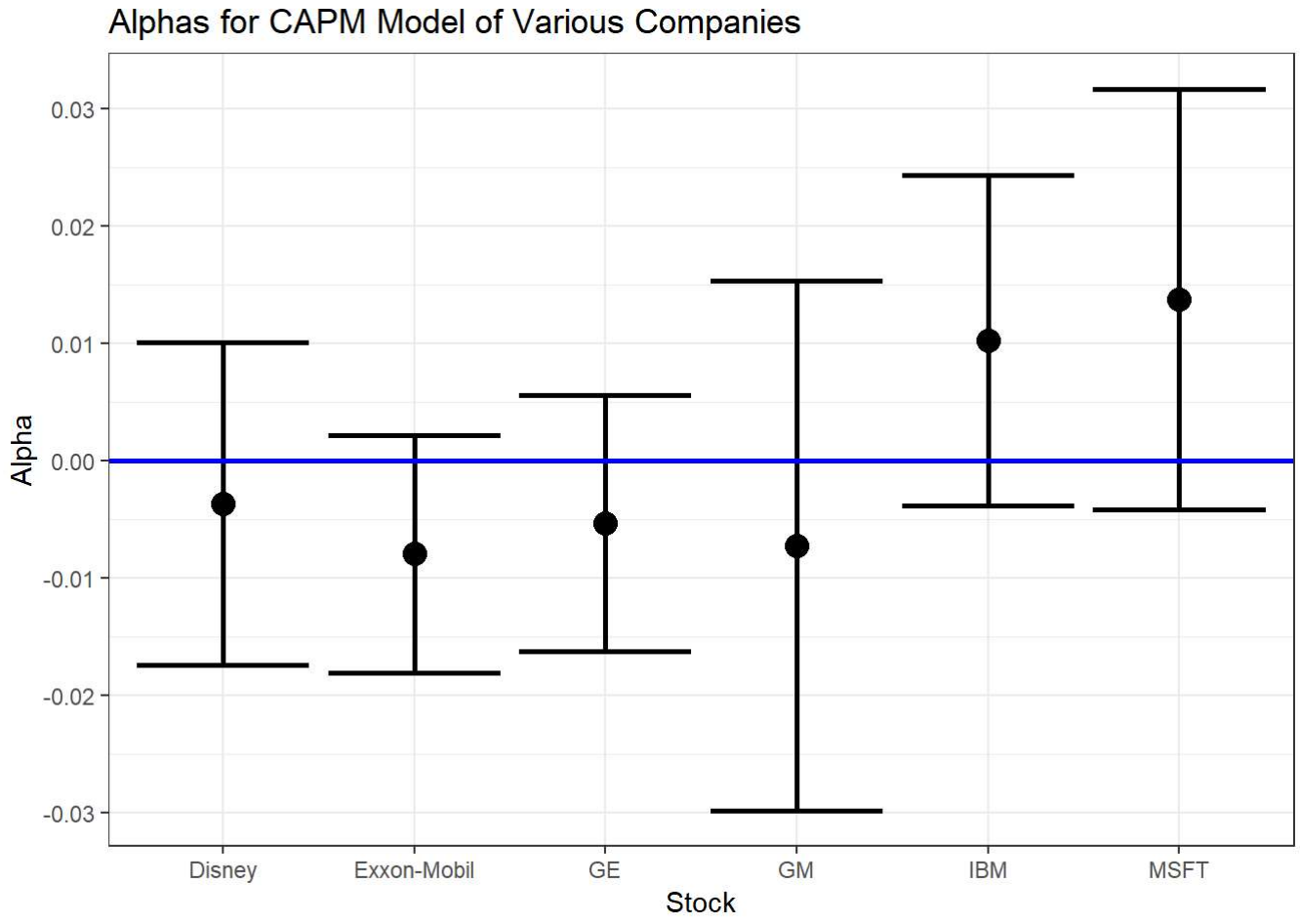
```
  scale_y_continuous(breaks = seq(0,2,0.1))+
  labs(
    title = "Betas for CAPM Model of Various Companies",
    x = "Stock",
    y = "Beta",
    caption = "Height of bars represent each stock's beta. Shown with horizontal blue line at be
ta = 1"
  )
```



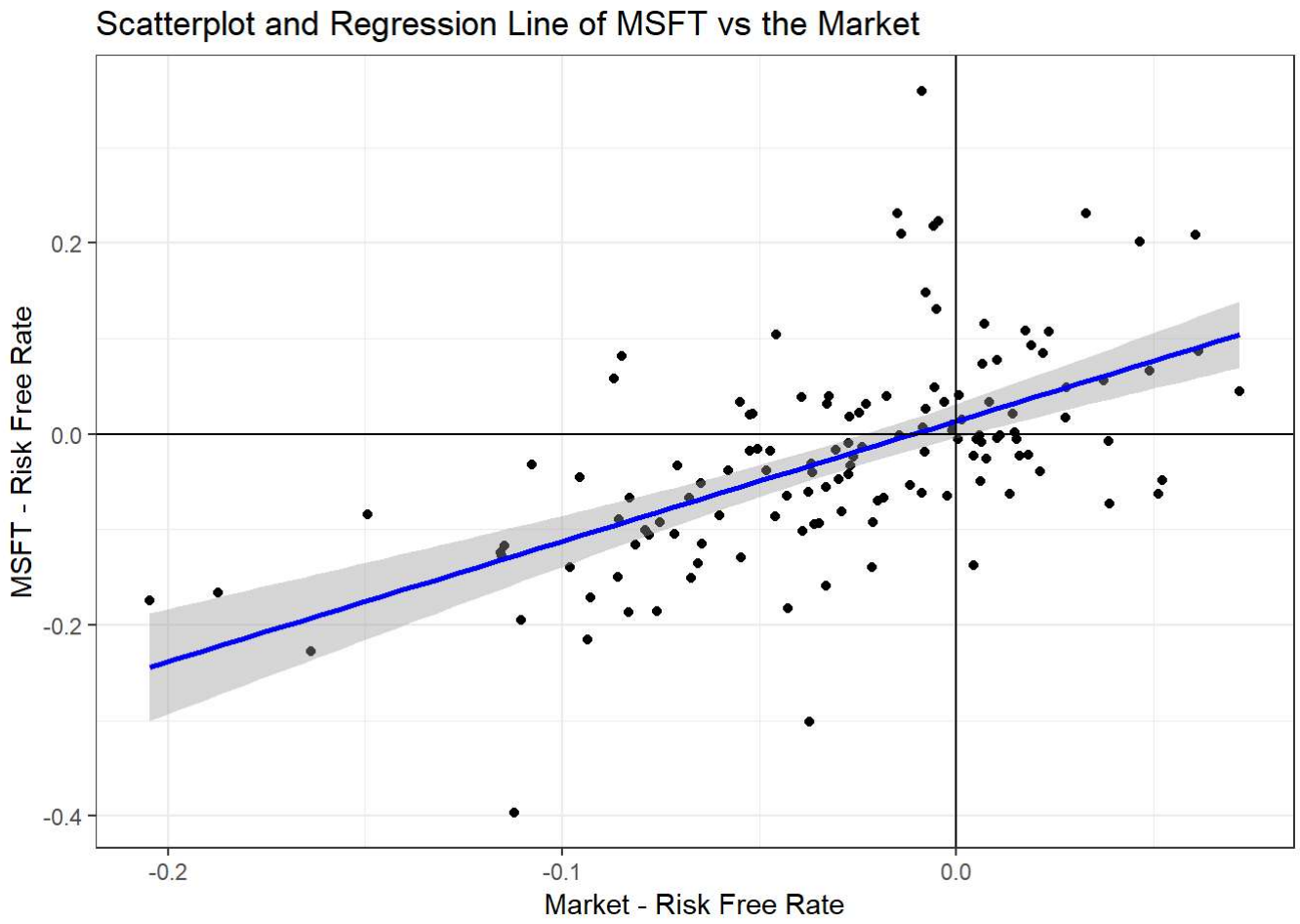
As shown in the above graph, there is a wide variety of betas for these companies. The most aggresive appears to be Microsoft at $\beta_{MSFT} \approx 1.286$, and the most defensive appears to be Exxon-Mobil at $\beta_{ExM} \approx 0.381$.

(b) Finance theory says that the intercept parameter α_j should be zero. Does this seem correct given your estimates? For the Microsoft Stock, plot the fitted regression line along with the data scatter.

```
#show alphas graphically
ggplot(alpha)+
  geom_point(aes(x = stock, y = alpha), size = 4)+
  geom_errorbar(aes(x = stock, ymax = confup, ymin = conflow), size = 1)+
  geom_hline(yintercept = 0, color = "blue", size = 1)+
  scale_y_continuous(breaks = seq(-0.05,0.05,0.01))+
  theme_bw()+
  labs(
    title = "Alphas for CAPM Model of Various Companies",
    x = "Stock",
    y = "Alpha"
  )
```



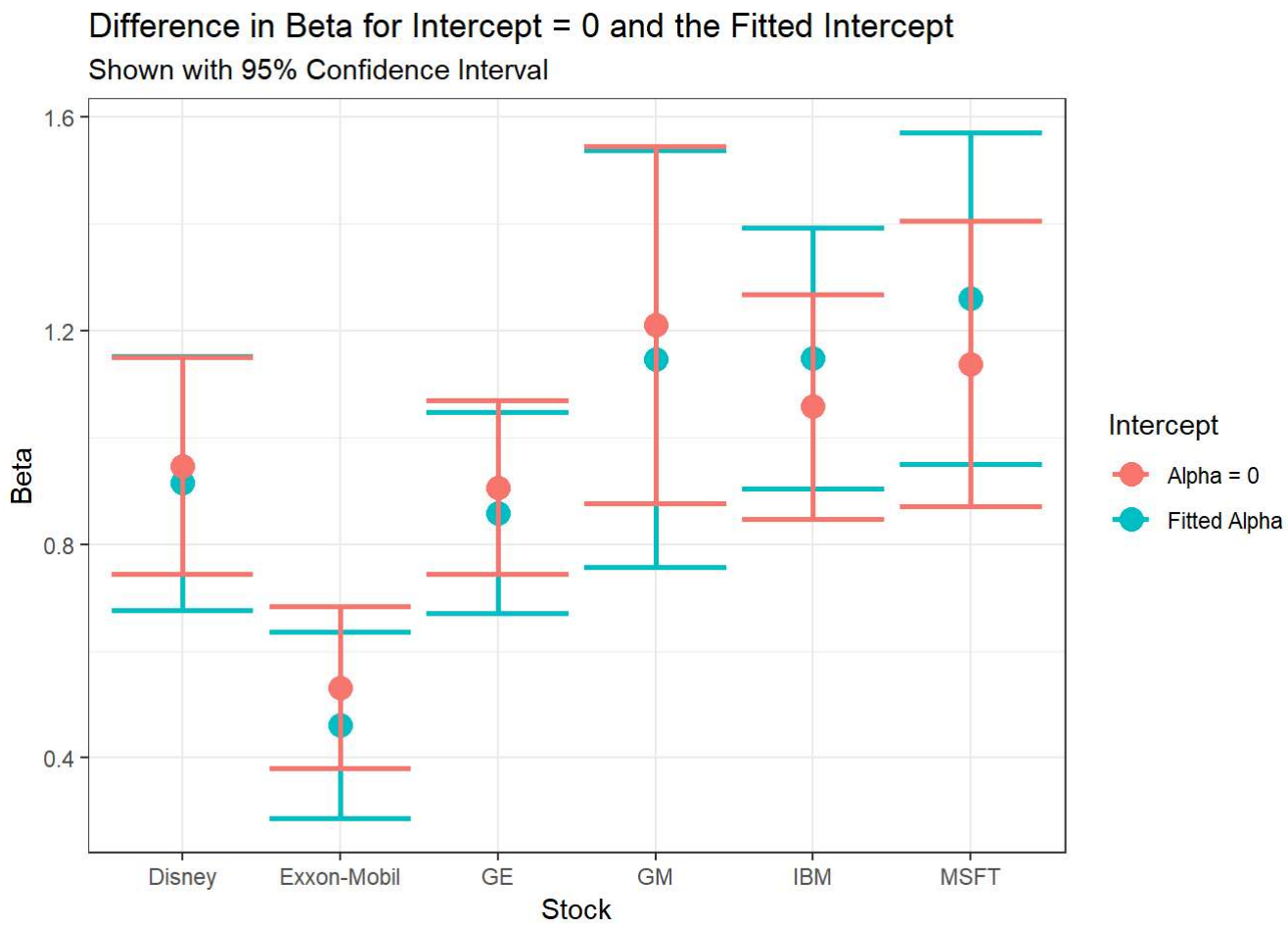

```
#present scatterplot of microsoft with fitted line
ggplot(market)+
  geom_point(aes(x = mktfree, y = msftfree))+
  stat_smooth(aes(x = mktfree, y = msftfree), method = "lm", color = "blue")+
  geom_vline(xintercept = 0, size = 0.5)+
  geom_hline(yintercept = 0, size = 0.5)+
  theme_bw()+
  labs(
    title = "Scatterplot and Regression Line of MSFT vs the Market",
    y = "MSFT - Risk Free Rate",
    x = "Market - Risk Free Rate"
  )
```



While the point estimates of alpha are non-zero, none of the stocks have a 95% confidence interval that excludes zero. Statistically speaking, we cannot conclude that the intercept is different from zero for any of the stocks. A scatterplot of the Microsoft stock and the fitted regression line shows that while the point estimate of the intercept is above zero, it is not far enough above zero to be considered different from zero, given the standard error.

(c) Estimate the model (coefficients and confidence intervals) for each firm under the assumption that $\alpha_j = 0$. Do the estimates of the beta values change much?

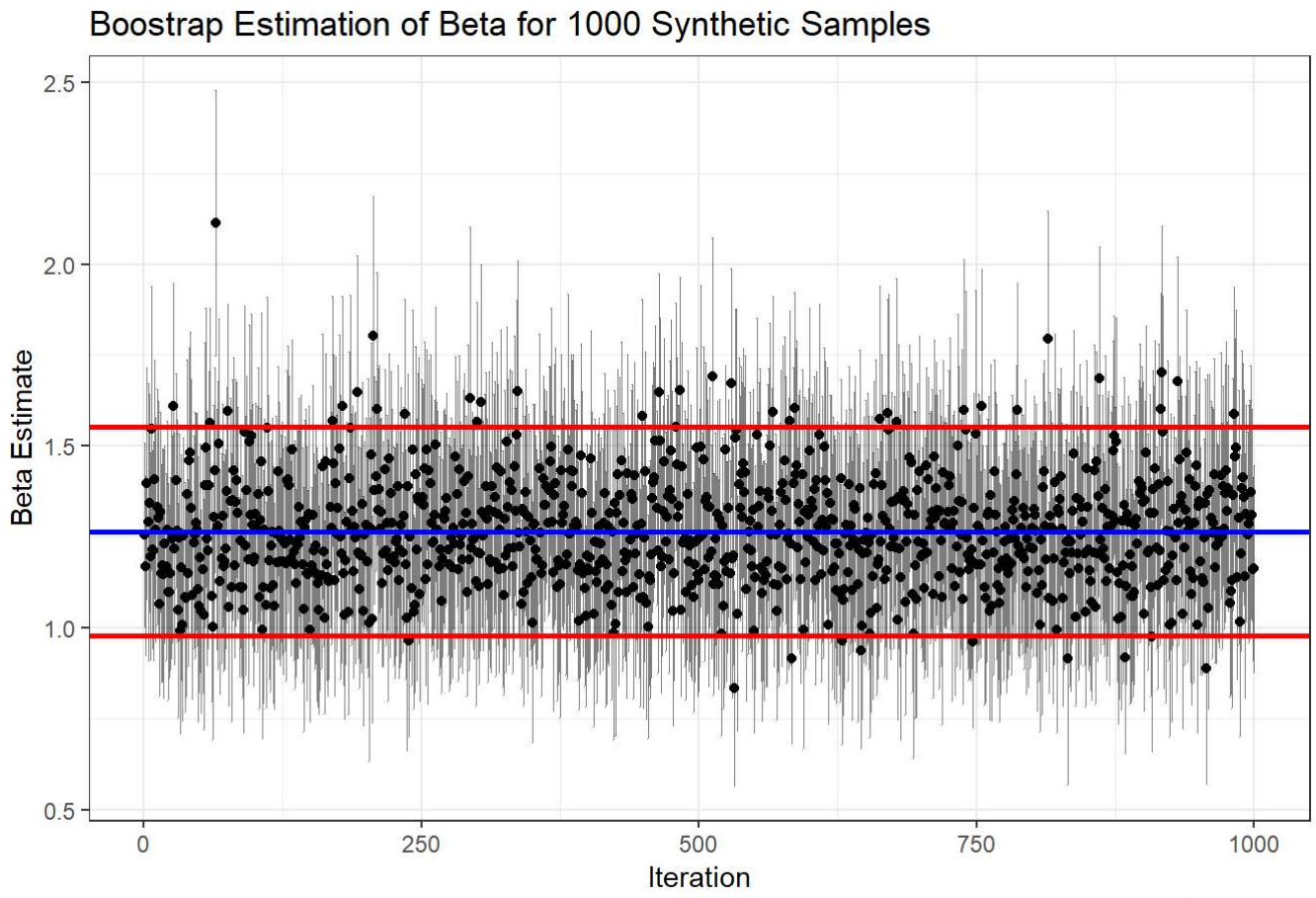
```
#estimate each stock
micInt = lm(msftfree ~ 0 + mktfree, data = market)
GEInt = lm(gefree ~ 0 + mktfree, data = market)
GMInt = lm(gmfree ~ 0 + mktfree, data = market)
IBMInt = lm(ibmfree ~ 0 + mktfree, data = market)
DisInt = lm(disfree ~ 0 + mktfree, data = market)
ExInt = lm(xomfree ~ 0 + mktfree, data = market)

#store stats of interest in df
betaInt = data.frame(
  stock = c("MSFT", "GE", "GM", "IBM", "Disney", "Exxon-Mobil"),
  beta = c(micInt[[1]][1], GEInt[[1]][1], GMInt[[1]][1], IBMInt[[1]][1], DisInt[[1]][1], ExInt[[
1]][1]),
  stde = c(summary(micInt)$coefficients[2], summary(GEInt)$coefficients[2], summary(GMInt)$coeff
icients[2], summary(IBMInt)$coefficients[2], summary(DisInt)$coefficients[2], summary(ExInt)$coe
fficients[2]),
  confup = NA,
  conflow = NA,
  Intercept = "Alpha = 0"
  )

#create 95% conf intervals
betaInt$confup = betaInt$beta + abs(qt(0.025,130))*betaInt$stde
betaInt$conflow = betaInt$beta - abs(qt(0.025,130))*betaInt$stde
beta$Intercept = "Fitted Alpha"
betaAll = rbind(beta, betaInt)

#compare betas
ggplot(betaAll)+
  geom_point(aes(x = stock, y = beta, color = Intercept), size = 4)+
  geom_errorbar(aes(x = stock, ymax = confup, ymin = conflow, color = Intercept), size = 1)+
  theme_bw()+
  labs(
    title = "Difference in Beta for Intercept = 0 and the Fitted Intercept",
    subtitle = "Shown with 95% Confidence Interval",
    x = "Stock",
    y = "Beta"
  )
```



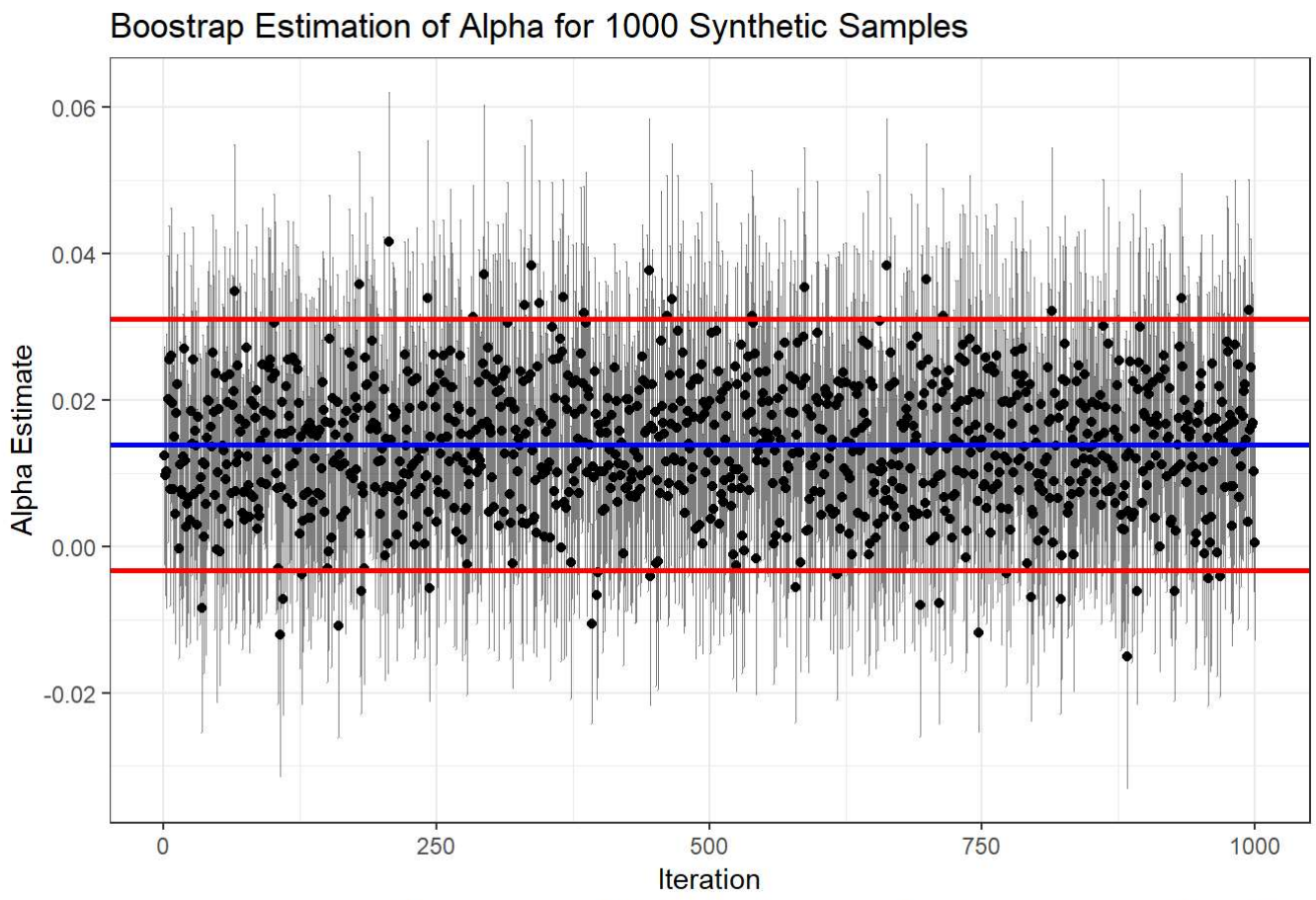
The beta values do not change appreciably when the intercept is forced to be zero. The beta values for the regressions with an intercept of zero fall within the 95% confidence interval of the beta values with a fitted intercept for ever stock considered, and vice versa, indicating that the beta values are not statistically different.

(d) Bootstrap Sampling: Generate 1000 synthetic samples from Microsoft by sampling with replacement, estimate the model parameters for each sample, and show separate plots of your estimated β and α across each realization. In addition, define the length of the respective confidence intervals by d_α and d_β, and plot these lengths across each realization. Comment on your results.

```
#set up initial data frame
bootparms = data.frame(
  iter = 1:1000, #1
  beta = NA,     #2
  bstde = NA,    #3
  bupper95 = NA, #4
  blower95 = NA, #5
  alpha = NA,    #6
  astde = NA,    #7
  aupper95 = NA, #8
  alower95 = NA  #9
  )

set.seed(12)

#Loop a bootstrap
for(i in 1:1000){
  boot = market[sample(c(1:nrow(market)), size = 132, replace = T), ]
  fit = lm(msftfree ~ mktfree, boot)
  bootparms[i, 2] = fit[[1]][2]
  bootparms[i, 3] = summary(fit)$coefficients[2, 2]
  bootparms[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  bootparms[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  bootparms[i, 6] = fit[[1]][1]
  bootparms[i, 7] = summary(fit)$coefficients[1, 2]
  bootparms[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  bootparms[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
}

#calculate confidence interval lengths
bootparms$dbeta = bootparms$bupper95 - bootparms$blower95
bootparms$dalpha = bootparms$aupper95 - bootparms$alower95

#plot the results for beta
ggplot(bootparms)+
  geom_point(aes(x = iter, y = beta))+
  geom_errorbar(aes(x = iter, ymax = bupper95, ymin = blower95), size = 0.1, alpha = 0.5)+
  geom_hline(yintercept = mean(bootparms$beta), color = "blue", size = 1)+
  geom_hline(yintercept = mean(bootparms$beta) + abs(qt(0.025, 130))*sqrt(var(bootparms$beta)),
 color = "red", size = 1)+
  geom_hline(yintercept = mean(bootparms$beta) - abs(qt(0.025, 130))*sqrt(var(bootparms$beta)),
 color = "red", size = 1)+
  theme_bw()+
  labs(
    title = "Boostrap Estimation of Beta for 1000 Synthetic Samples",
    x = "Iteration",
    y = "Beta Estimate",
    caption = "Bootstrap beta represented by blue line. Bootstrap confidence interval reperesent
ed by red lines."
  )
```

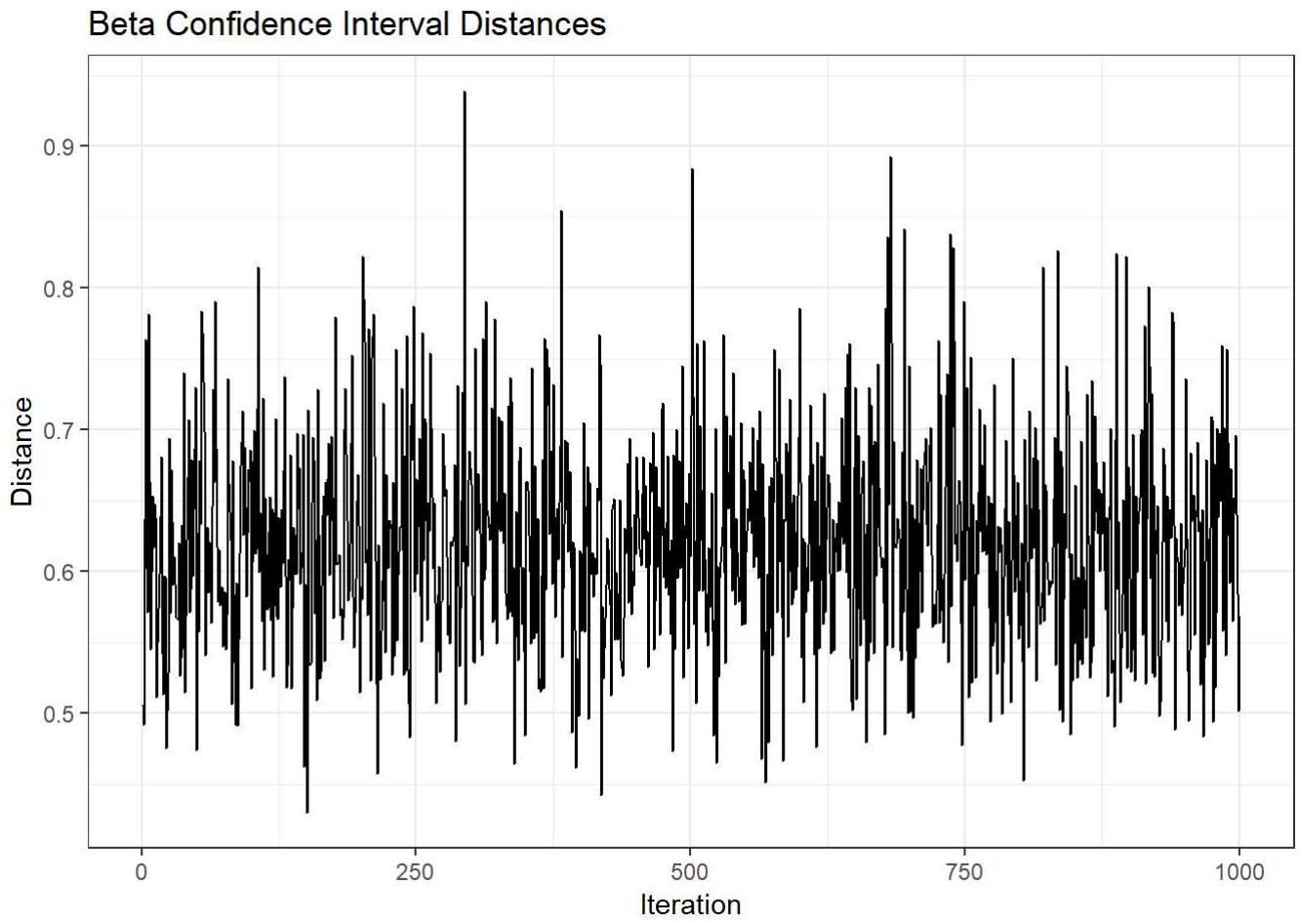



```
#plot the results for alpha
ggplot(bootparms)+
  geom_point(aes(x = iter, y = alpha))+
  geom_errorbar(aes(x = iter, ymax = aupper95, ymin = alower95), size = 0.1, alpha = 0.5)+
  geom_hline(yintercept = mean(bootparms$alpha), color = "blue", size = 1)+
  geom_hline(yintercept = mean(bootparms$alpha) + abs(qt(0.025, 130))*sqrt(var(bootparms$alph
a)), color = "red", size = 1)+
  geom_hline(yintercept = mean(bootparms$alpha) - abs(qt(0.025, 130))*sqrt(var(bootparms$alph
a)), color = "red", size = 1)+
  theme_bw()+
  labs(
    title = "Boostrap Estimation of Alpha for 1000 Synthetic Samples",
    x = "Iteration",
    y = "Alpha Estimate",
    caption = "Bootstrap alpha represented by blue line. Bootstrap confidence interval reperesen
ted by red lines."
  )
```

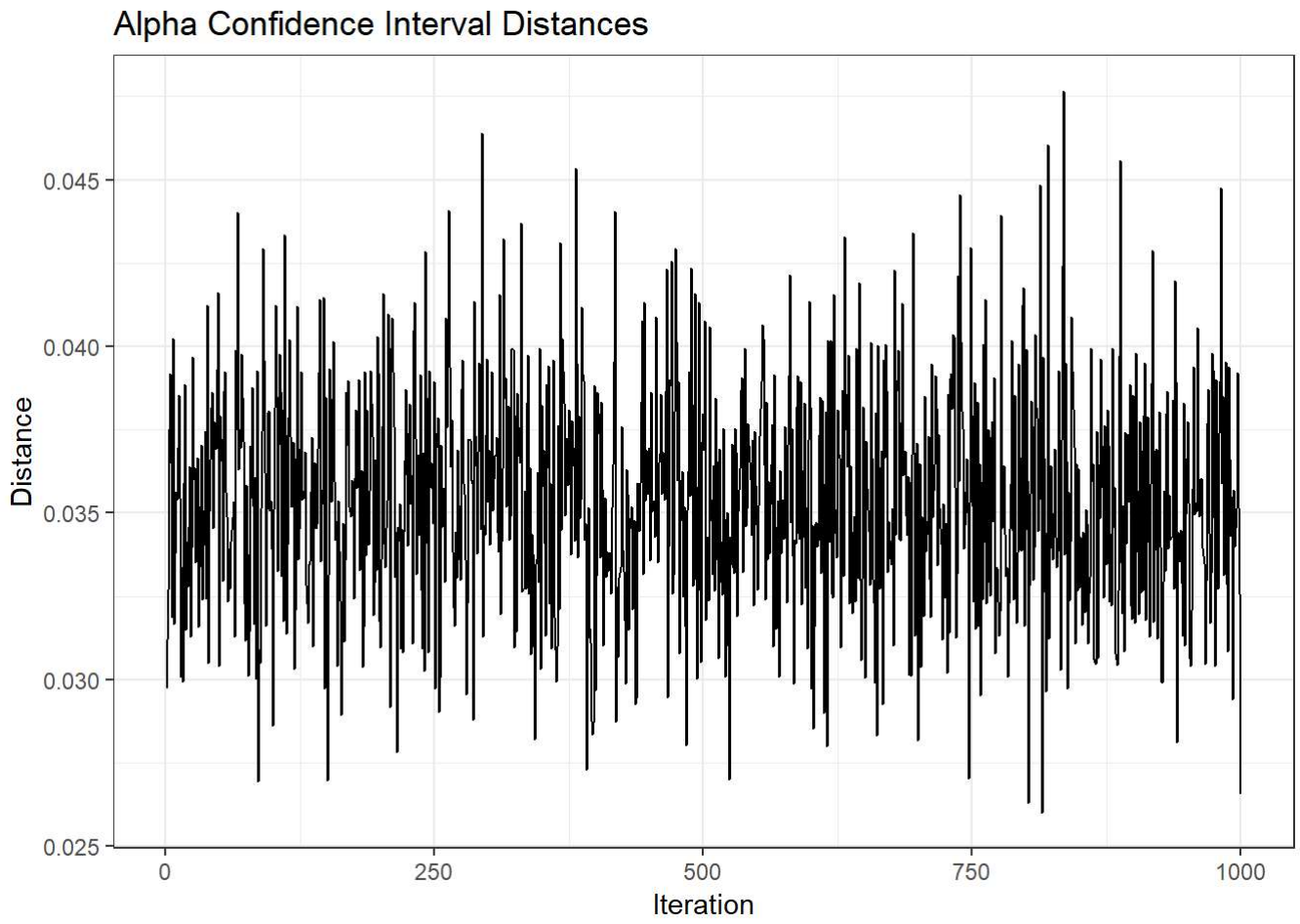


Bootstrap alpha represented by blue line. Bootstrap confidence interval reperesented by red lines.

```
ggplot(bootparms)+
  geom_line(aes(x = iter, y = dbeta))+
  theme_bw()+
  labs(
    title = "Beta Confidence Interval Distances",
    x = "Iteration",
    y = "Distance"
  )
```




```
ggplot(bootparms)+
  geom_line(aes(x = iter, y = dalpha))+
  theme_bw()+
  labs(
    title = "Alpha Confidence Interval Distances",
    x = "Iteration",
    y = "Distance"
  )
```




```
c(max(bootparms$dbeta), min(bootparms$dbeta))
```

```
## [1] 0.9390846 0.4303186
```

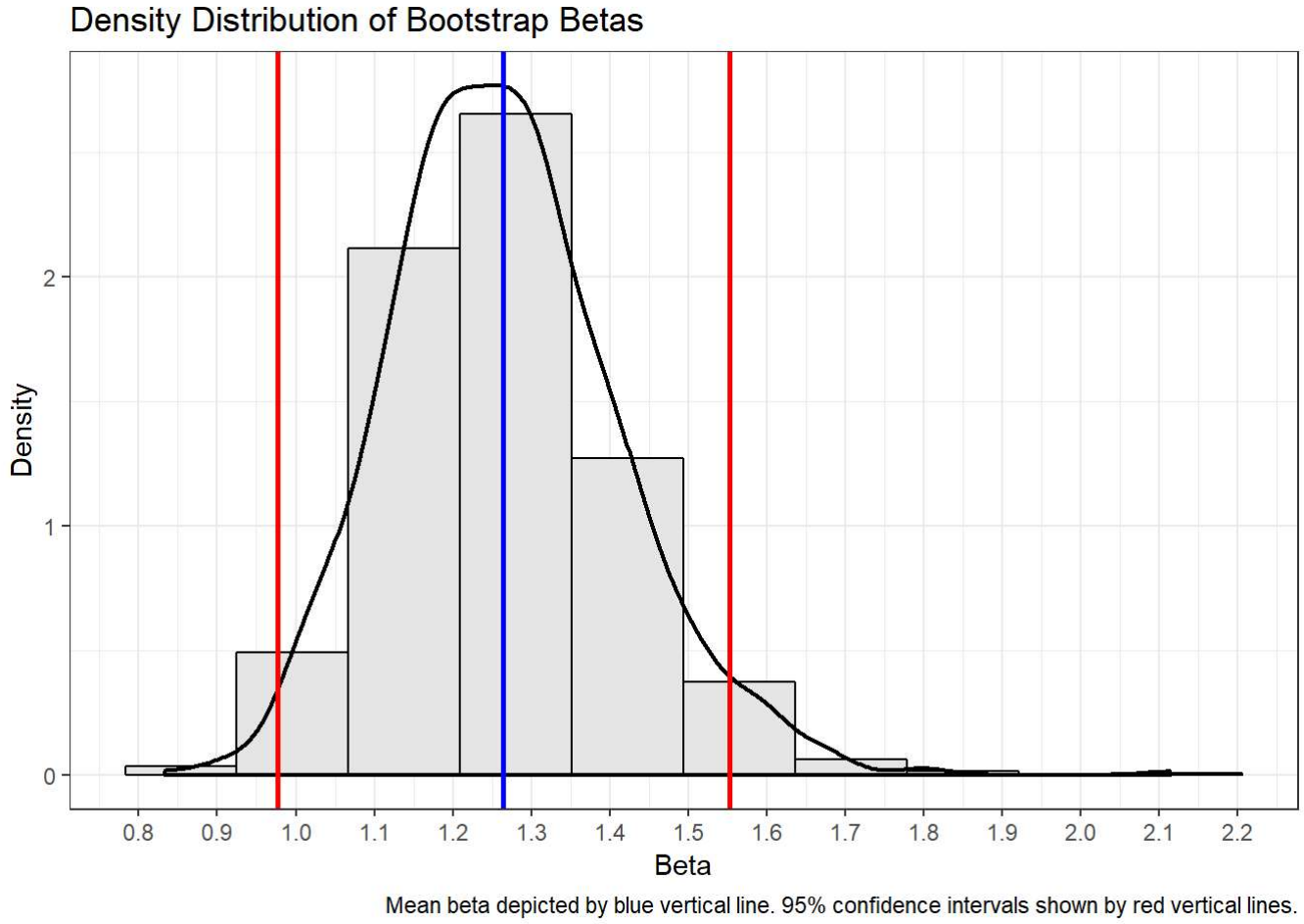
```
c(max(bootparms$dalpha), min(bootparms$dalpha))
```

```
## [1] 0.04766828 0.02602310
```

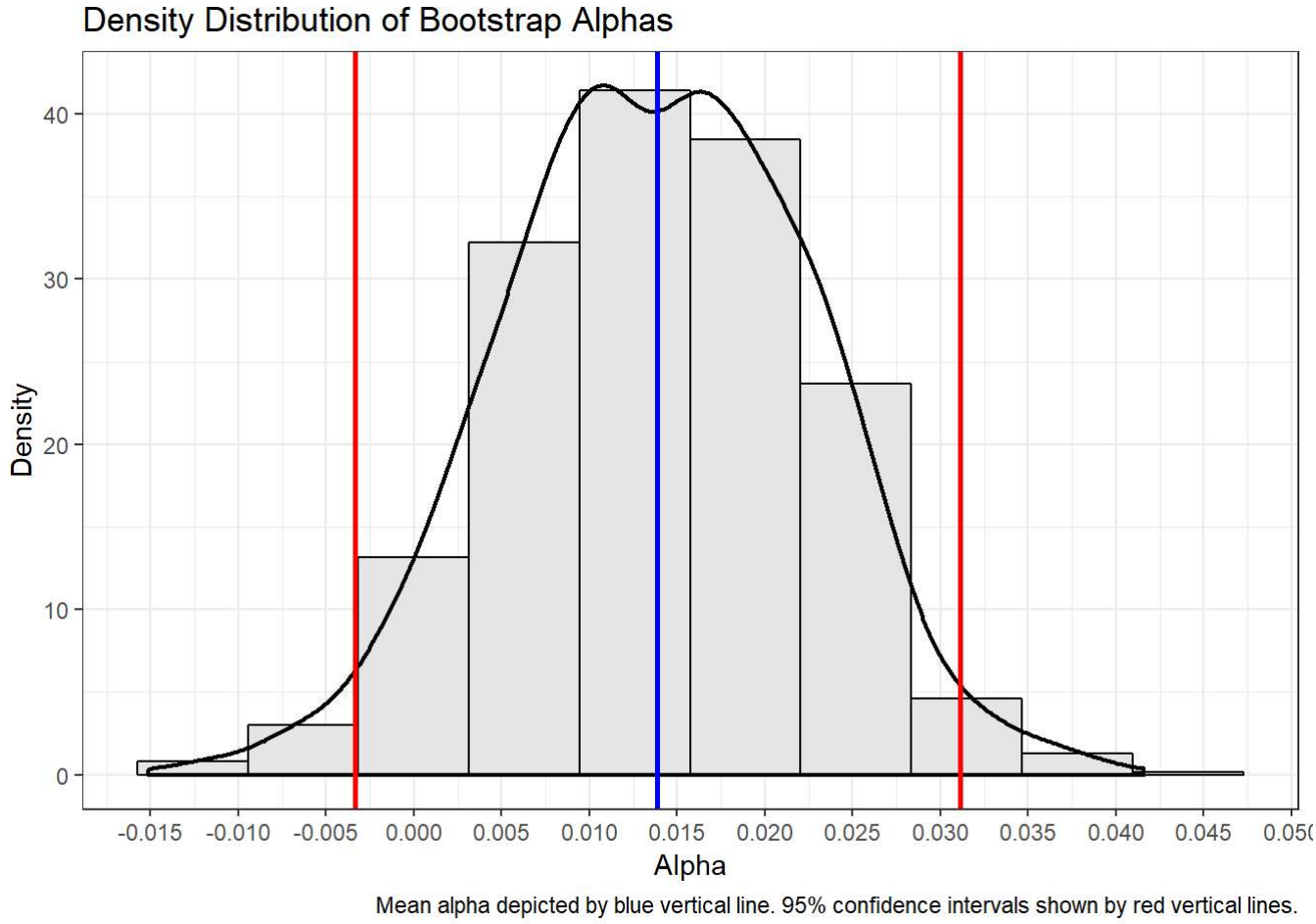
The value of both beta and alpha seem to vary pretty widely. There is a lot of noise when looking at the plots for beta and alpha and there is a lot of noise in the plots depicting the distances of the confidence intervals. The distances for beta can vary from 0.414 to 0.839 and given that we are looking at percentage returns less risk free returns, this is a very large variation. The distances for alpha do not vary as much as for beta, but still very quite a bit for an intercept that is theoretically supposed to be zero.

(e) Compute the mean and volatility of your estimates from part (d) for $\hat{\beta}$, $\hat{\mathbf{d}}_\beta$, $\hat{\alpha}$, and $\hat{\mathbf{d}}_\alpha$, and plot a histogram (including the respective density curve) for each one of your estimates

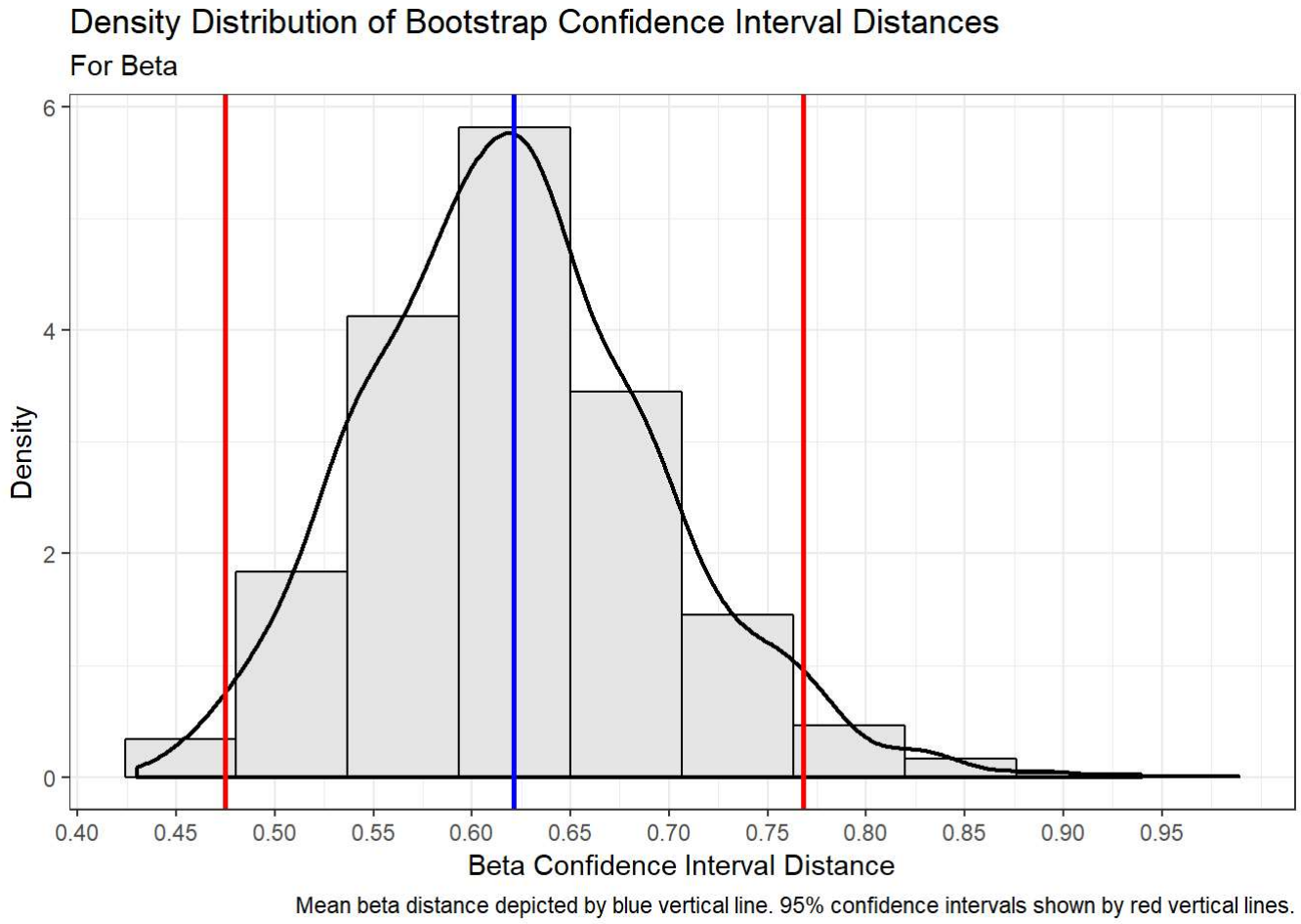
```
ggplot(bootparms)+
  geom_histogram(aes(x = beta, y = ..density..), color = "black", fill = "grey90", bins = (1+log
2(nrow(bootparms))))+
  geom_density(aes(x = beta), size = 0.705)+
  geom_vline(xintercept = mean(bootparms$beta), color = "blue", size = 1)+
  geom_vline(xintercept = mean(bootparms$beta) + abs(qt(0.025, 130))*sqrt(var(bootparms$beta)),
 color = "red", size = 1)+
  geom_vline(xintercept = mean(bootparms$beta) - abs(qt(0.025, 130))*sqrt(var(bootparms$beta)),
 color = "red", size = 1)+
  scale_x_continuous(breaks = seq(0.5,2.5, 0.1))+
  theme_bw()+
  labs(
    title = "Density Distribution of Bootstrap Betas",
    x = "Beta",
    y = "Density",
    caption = "Mean beta depicted by blue vertical line. 95% confidence intervals shown by red v
ertical lines."
  )
```



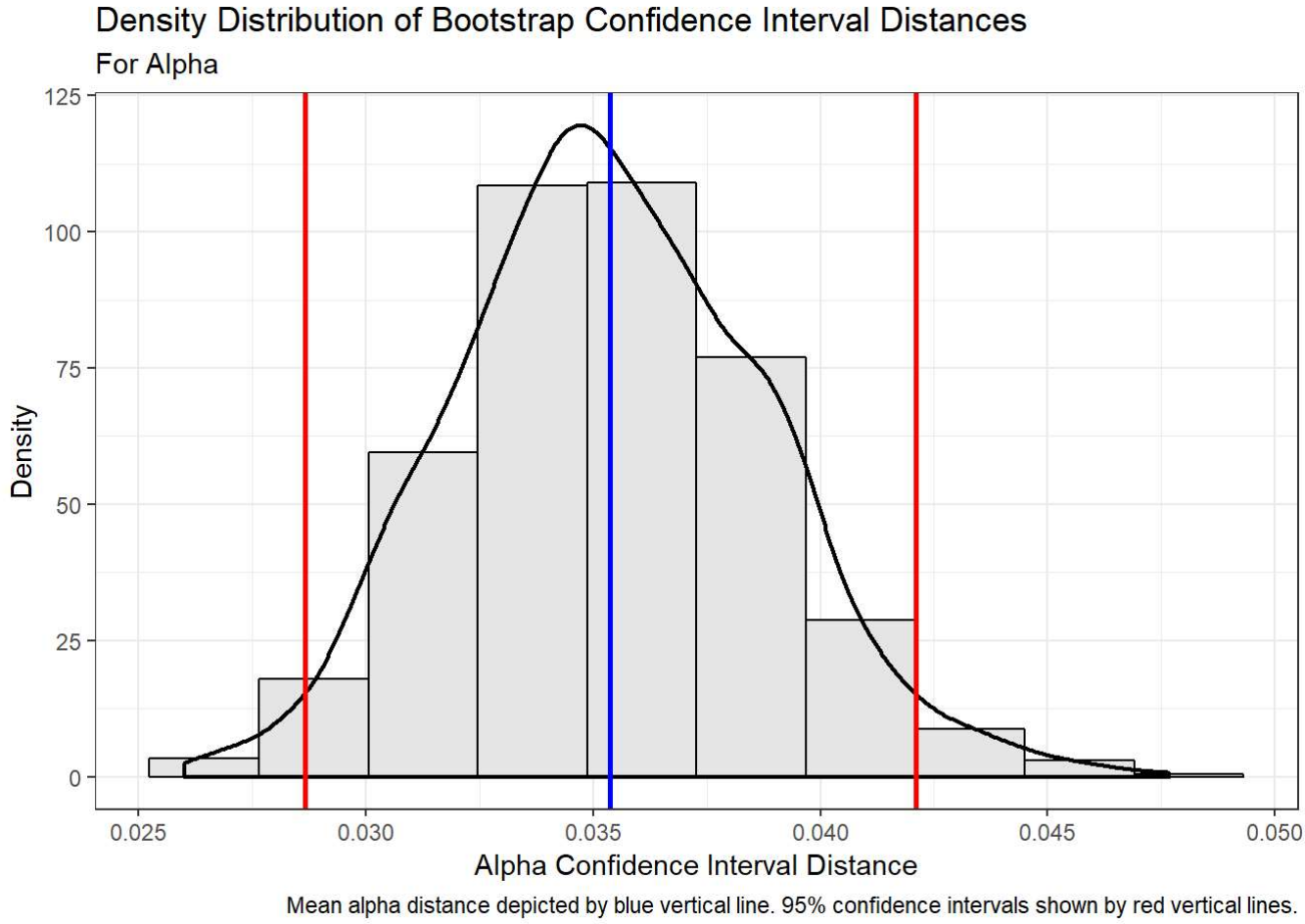
```
ggplot(bootparms)+
  geom_histogram(aes(x = alpha, y = ..density..), color = "black", fill = "grey90", bins = (1+lo
g2(nrow(bootparms))))+
  geom_density(aes(x = alpha), size = 0.705)+
  geom_vline(xintercept = mean(bootparms$alpha), color = "blue", size = 1)+
  geom_vline(xintercept = mean(bootparms$alpha) + abs(qt(0.025, 130))*sqrt(var(bootparms$alph
a)), color = "red", size = 1)+
  geom_vline(xintercept = mean(bootparms$alpha) - abs(qt(0.025, 130))*sqrt(var(bootparms$alph
a)), color = "red", size = 1)+
  scale_x_continuous(breaks = seq(-0.02,0.05, 0.005))+
  theme_bw()+
  labs(
    title = "Density Distribution of Bootstrap Alphas",
    x = "Alpha",
    y = "Density",
    caption = "Mean alpha depicted by blue vertical line. 95% confidence intervals shown by red
 vertical lines."
  )
```



```
ggplot(bootparms)+
  geom_histogram(aes(x = dbeta, y = ..density..), color = "black", fill = "grey90", bins = (1+lo
g2(nrow(bootparms))))+
  geom_density(aes(x = dbeta), size = 0.705)+
  geom_vline(xintercept = mean(bootparms$dbeta), color = "blue", size = 1)+
  geom_vline(xintercept = mean(bootparms$dbeta) + abs(qt(0.025, 130))*sqrt(var(bootparms$dbet
a)), color = "red", size = 1)+
  geom_vline(xintercept = mean(bootparms$dbeta) - abs(qt(0.025, 130))*sqrt(var(bootparms$dbet
a)), color = "red", size = 1)+
  scale_x_continuous(breaks = seq(-0.5,0.95, 0.05))+
  theme_bw()+
  labs(
    title = "Density Distribution of Bootstrap Confidence Interval Distances",
    subtitle = "For Beta",
    x = "Beta Confidence Interval Distance",
    y = "Density",
    caption = "Mean beta distance depicted by blue vertical line. 95% confidence intervals shown
 by red vertical lines."
  )
```



```
ggplot(bootparms)+
  geom_histogram(aes(x = dalpha, y = ..density..), color = "black", fill = "grey90", bins = (1+l
og2(nrow(bootparms))))+
  geom_density(aes(x = dalpha), size = 0.705)+
  geom_vline(xintercept = mean(bootparms$dalpha), color = "blue", size = 1)+
  geom_vline(xintercept = mean(bootparms$dalpha) + abs(qt(0.025, 130))*sqrt(var(bootparms$dalph
a)), color = "red", size = 1)+
  geom_vline(xintercept = mean(bootparms$dalpha) - abs(qt(0.025, 130))*sqrt(var(bootparms$dalph
a)), color = "red", size = 1)+
  scale_x_continuous(breaks = seq(0.025, 0.05, 0.005))+
  theme_bw()+
  labs(
    title = "Density Distribution of Bootstrap Confidence Interval Distances",
    subtitle = "For Alpha",
    x = "Alpha Confidence Interval Distance",
    y = "Density",
    caption = "Mean alpha distance depicted by blue vertical line. 95% confidence intervals show
n by red vertical lines."
  )
```

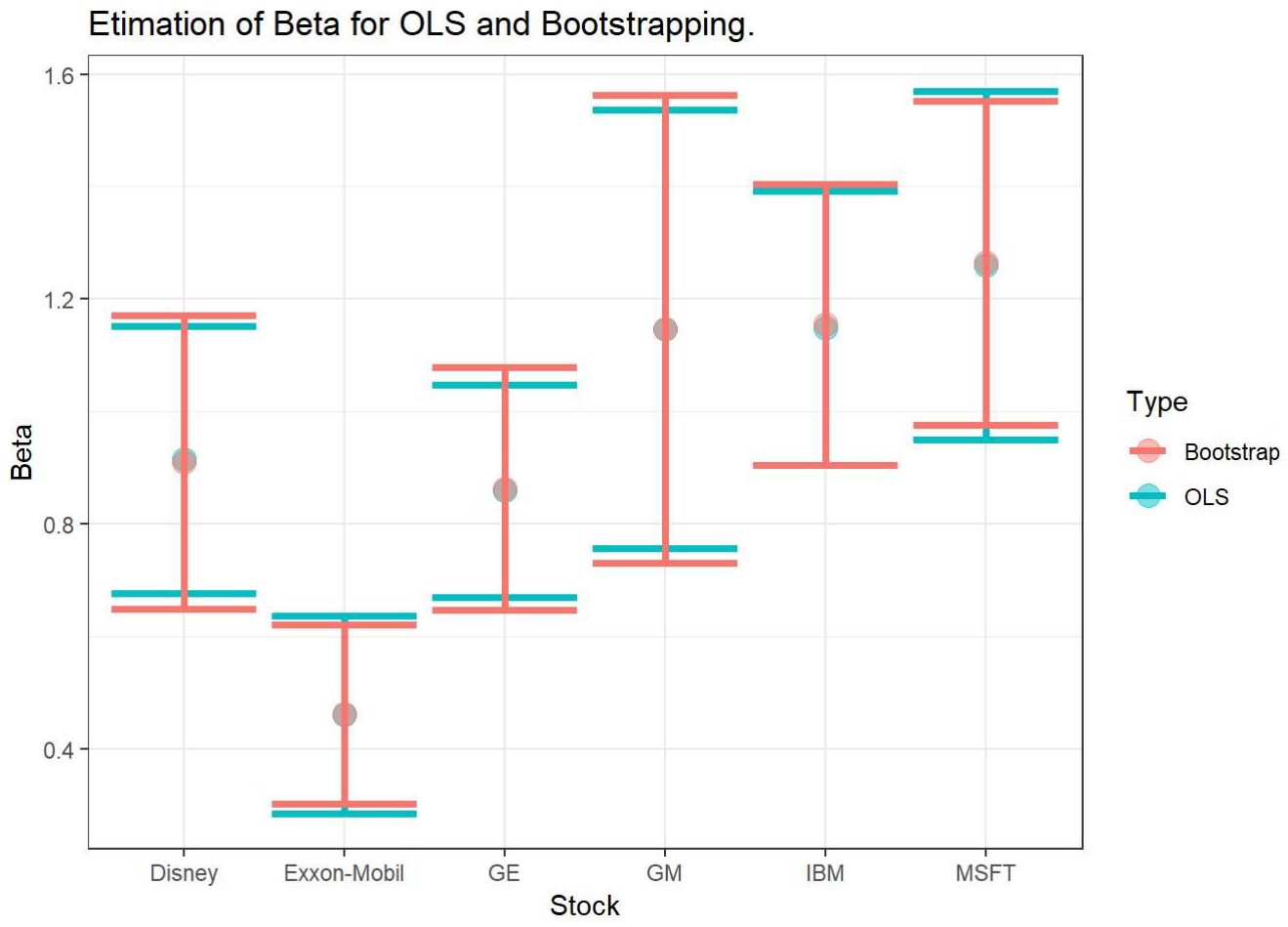


(f) How do your results from part (e) compare against those from part (a)? Which one do you trust more and why?

```
a2e = beta
a2e = a2e[-6]
a2e$Type = "OLS"

a2e1 = data.frame(stock = beta[1], beta = NA, stde = NA, confup = NA, conflow = NA, Type = "Boot
strap")

msftboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha
 = NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)
geboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha =
 NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)
gmboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha =
 NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)
ibmboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha
 = NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)
disboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha
 = NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)
exboots = data.frame(iter = 1:1000, beta = NA, bstde = NA, bupper95 = NA, blower95 = NA, alpha =
 NA, astde = NA, aupper95 = NA, alower95 = NA, dbeta = NA, dalpha = NA)

set.seed(12)

#Loop a bootstrap
for(i in 1:1000){
  boot = market[sample(c(1:nrow(market)), size = 132, replace = T), ]

  fit = lm(msftfree ~ mktfree, boot)
  msftboots[i, 2] = fit[[1]][2]
  msftboots[i, 3] = summary(fit)$coefficients[2, 2]
  msftboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  msftboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  msftboots[i, 6] = fit[[1]][1]
  msftboots[i, 7] = summary(fit)$coefficients[1, 2]
  msftboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  msftboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])

  fit = lm(gefree ~ mktfree, boot)
  geboots[i, 2] = fit[[1]][2]
  geboots[i, 3] = summary(fit)$coefficients[2, 2]
  geboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  geboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  geboots[i, 6] = fit[[1]][1]
  geboots[i, 7] = summary(fit)$coefficients[1, 2]
  geboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  geboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])

  fit = lm(gmfree ~ mktfree, boot)
  gmboots[i, 2] = fit[[1]][2]
  gmboots[i, 3] = summary(fit)$coefficients[2, 2]
  gmboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  gmboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  gmboots[i, 6] = fit[[1]][1]
  gmboots[i, 7] = summary(fit)$coefficients[1, 2]
```

```
  gmboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  gmboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])

  fit = lm(ibmfree ~ mktfree, boot)
  ibmboots[i, 2] = fit[[1]][2]
  ibmboots[i, 3] = summary(fit)$coefficients[2, 2]
  ibmboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  ibmboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  ibmboots[i, 6] = fit[[1]][1]
  ibmboots[i, 7] = summary(fit)$coefficients[1, 2]
  ibmboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  ibmboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])

  fit = lm(disfree ~ mktfree, boot)
  disboots[i, 2] = fit[[1]][2]
  disboots[i, 3] = summary(fit)$coefficients[2, 2]
  disboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  disboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  disboots[i, 6] = fit[[1]][1]
  disboots[i, 7] = summary(fit)$coefficients[1, 2]
  disboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  disboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])

  fit = lm(xomfree ~ mktfree, boot)
  exboots[i, 2] = fit[[1]][2]
  exboots[i, 3] = summary(fit)$coefficients[2, 2]
  exboots[i, 4] = fit[[1]][2] + abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  exboots[i, 5] = fit[[1]][2] - abs(qt(0.025, 130)*summary(fit)$coefficients[2, 2])
  exboots[i, 6] = fit[[1]][1]
  exboots[i, 7] = summary(fit)$coefficients[1, 2]
  exboots[i, 8] = fit[[1]][1] + abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
  exboots[i, 9] = fit[[1]][1] - abs(qt(0.025, 130)*summary(fit)$coefficients[1, 2])
}

#rm(GE, GEInt, GM, GMInt, IBM, IBMInt, micInt, micsoft)

a2e1[1,2] = mean(msftboots$beta)
a2e1[1,3] = sqrt(var(msftboots$beta))
a2e1[1,4] = mean(msftboots$beta) + abs(qt(0.025, 130))*sqrt(var(msftboots$beta))
a2e1[1,5] = mean(msftboots$beta) - abs(qt(0.025, 130))*sqrt(var(msftboots$beta))

a2e1[2,2] = mean(geboots$beta)
a2e1[2,3] = sqrt(var(geboots$beta))
a2e1[2,4] = mean(geboots$beta) + abs(qt(0.025, 130))*sqrt(var(geboots$beta))
a2e1[2,5] = mean(geboots$beta) - abs(qt(0.025, 130))*sqrt(var(geboots$beta))

a2e1[3,2] = mean(gmboots$beta)
a2e1[3,3] = sqrt(var(gmboots$beta))
a2e1[3,4] = mean(gmboots$beta) + abs(qt(0.025, 130))*sqrt(var(gmboots$beta))
a2e1[3,5] = mean(gmboots$beta) - abs(qt(0.025, 130))*sqrt(var(gmboots$beta))

a2e1[4,2] = mean(ibmboots$beta)
a2e1[4,3] = sqrt(var(ibmboots$beta))
a2e1[4,4] = mean(ibmboots$beta) + abs(qt(0.025, 130))*sqrt(var(ibmboots$beta))
```

```
a2e1[4,5] = mean(ibmboots$beta) - abs(qt(0.025, 130))*sqrt(var(ibmboots$beta))

a2e1[5,2] = mean(disboots$beta)
a2e1[5,3] = sqrt(var(disboots$beta))
a2e1[5,4] = mean(disboots$beta) + abs(qt(0.025, 130))*sqrt(var(disboots$beta))
a2e1[5,5] = mean(disboots$beta) - abs(qt(0.025, 130))*sqrt(var(disboots$beta))

a2e1[6,2] = mean(exboots$beta)
a2e1[6,3] = sqrt(var(exboots$beta))
a2e1[6,4] = mean(exboots$beta) + abs(qt(0.025, 130))*sqrt(var(exboots$beta))
a2e1[6,5] = mean(exboots$beta) - abs(qt(0.025, 130))*sqrt(var(exboots$beta))

a2e = rbind(a2e, a2e1)

rm(a2e1, disboots, DisInt, Disney, exboots, ExInt, fit, geboots, ExMobil, gmboots, ibmboots, msf
tboots)

ggplot(a2e)+
  geom_point(aes(x = stock, y = beta, color = Type), size = 4, alpha = 0.5)+
  geom_errorbar(aes(x = stock, ymax = confup, ymin = conflow, color = Type), size = 1.25)+
  theme_bw()+
  labs(
    title = "Etimation of Beta for OLS and Bootstrapping.",
    x = "Stock",
    y = "Beta"
  )
```



The differences between the standard OLS betas and the bootsrap betas are very small. Both the point estimates and the 95% confidence intervals are very similar for both types of estimation. As far as which is more trustworthy, bootstrapping is a much more rigorous way of estimating. By resampling from our initial sample repeatedly, we are better able to guage the stability of our estimates through a simulation of more samples. In many respects, bootstrapping gives us more information than OLS does and more information is always better.

Question 4: Sequential Bayesian Learning

You will need to implement this in R using loops. Assume we are given the following facts:

- 1% of women aged 40 have breast cancer.
- A mammography test has a 99% success rate, and a 10% false alarm rate (i.e., 10% of the time, the test will return positive for having cancer when the patient does not actually have cancer).

(a) Given the above, a woman aged 40 receives a positive mammography test. What is the probability that she actually has cancer? Let C+ = Cancer presnt and T+ = Positive mammography test, you need to find P(C+|T+).

```
#set up parameters for the Bayesian equation
p_cancer =0.01
p_post_cancer =0.99
p_post_ncancer=0.1
p_ncancer=0.99

prior = p_cancer
likelihood = p_post_cancer
norm_coeff = (prior*likelihood)+(p_post_ncancer*(1-prior))

posterior = (prior*likelihood)/norm_coeff              #compute posterior

posterior
```

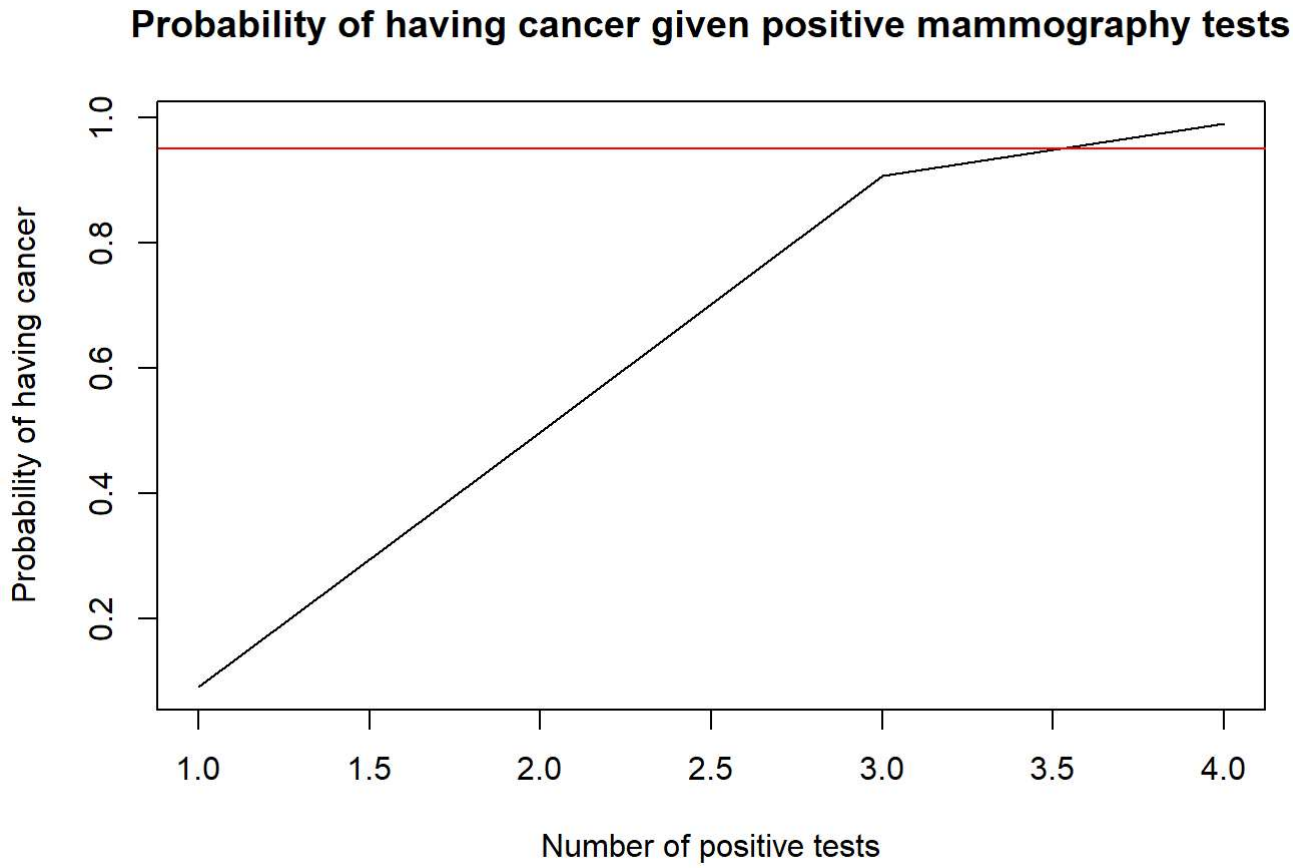
```
## [1] 0.09090909
```

Therefore, the probability of having cancer after receiving a positive test is about 9.09%.

(b) Assume the same woman from (a), wants to get another opinion, and then another, and so on, but that every time she gets tested, the results are the same as the first one. After how many trials, will P(C+|T+)>95%? Show a plot of P(C+|T+) vs Trial Number, and comment on your finding.

```
#set up parameters
p_cancer =0.01
p_post_cancer =0.99
p_post_ncancer=0.1
p_ncancer=0.99

prior = p_cancer
likelihood = p_post_cancer
norm_coeff = (prior*likelihood)+(p_post_ncancer*(1-prior))

#setup while loop
posterior=0
trial=1
post=c()
while(posterior<=0.95){
  posterior = (prior*likelihood)/norm_coeff
  post[trial]=posterior
  prior = posterior
  norm_coeff = (prior*likelihood)+(p_post_ncancer*(1-prior))

  if(posterior<=0.95){
    trial=trial+1
  }

}

plot.new()

plot(1:trial,post,type="l",
     main = "Probability of having cancer given positive mammography tests",
     xlab="Number of positive tests", ylab="Probability of having cancer")
abline(h=0.95,type=2,col="red")
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...):
## graphical parameter "type" is obsolete
```

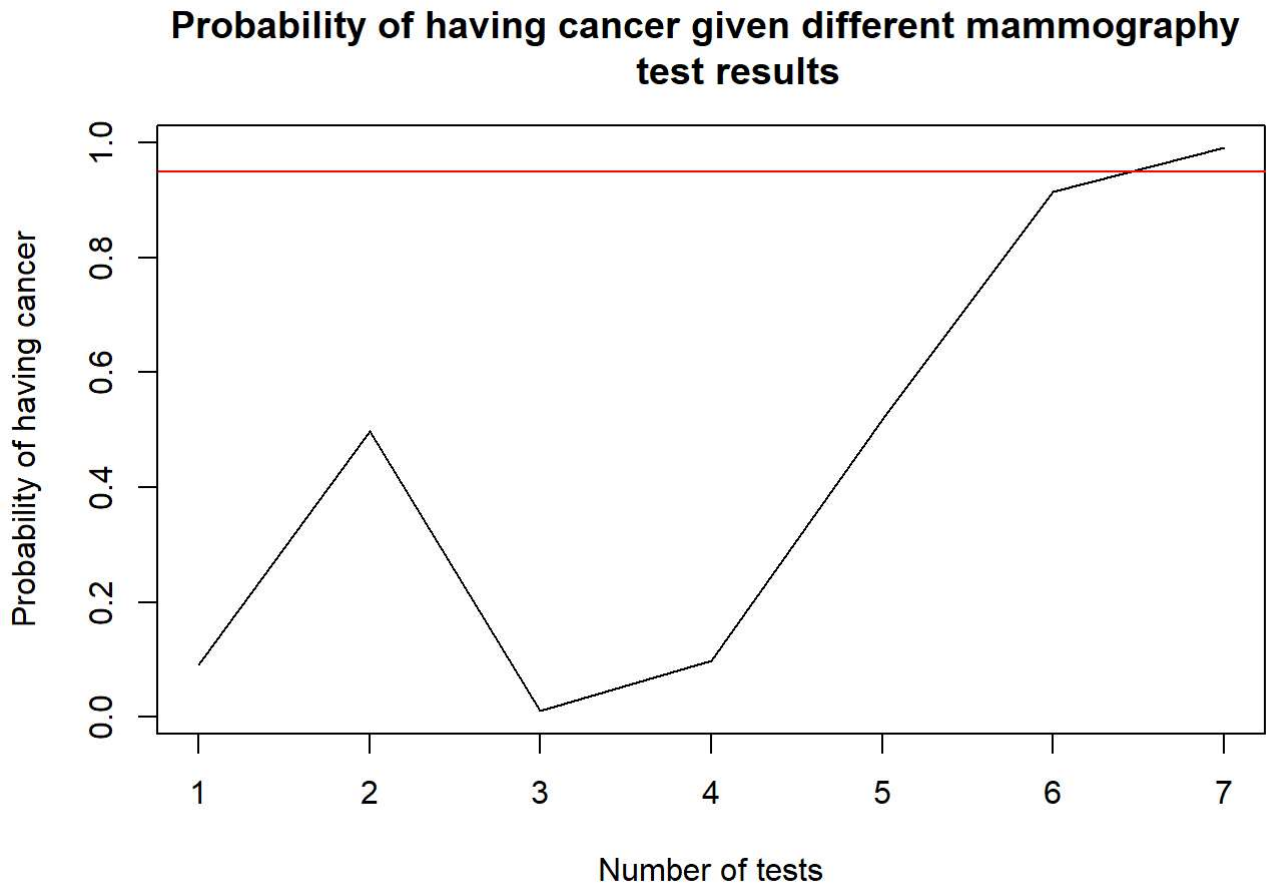


From the plot, we see that the probability of having cancer is greater than 95% after 4 trials. We notice that the probability of having cancer increases quickly with each positive test.

(c) In on Trial 3, the test results show negative for breast cancer, but all other tests show positive, how would this affect P(C+|T+) found in (b)?

```
#setup parameters
p_cancer =0.01
p_post_cancer =0.99
p_post_ncancer=0.1
p_ncancer=0.99

p_negt_cancer = 0.01
p_negt_ncancer = 0.9

prior = p_cancer
likelihood = p_post_cancer
likelihood2 =p_negt_cancer
norm_coeff = (prior*likelihood)+(p_post_ncancer*(1-prior))

#setup while loop
posterior=0
trial=1
post=c()
while(posterior<=0.95){
  norm_coeff = (prior*likelihood)+(p_post_ncancer*(1-prior))
  if(trial!=3){
    posterior = (prior*likelihood)/norm_coeff


  }else{
    norm_coeff = (prior*likelihood2)+(p_negt_ncancer*(1-prior))
    posterior = (prior*likelihood2)/norm_coeff

  }
  post[trial]=posterior
  prior = posterior

  if(posterior<=0.95){
    trial=trial+1
  }

}

plot.new()

plot(1:trial,post,type="l",
     main = "Probability of having cancer given different mammography
     test results",
     xlab="Number of tests", ylab="Probability of having cancer")
abline(h=0.95,type=2,col="red")
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...):
## graphical parameter "type" is obsolete
```



From this plot we can see that it takes seven trials in order to get above the 95% probability of having cancer. This means that the one negative test caused it to take longer for the probability to increase past 95%.