Econ 424 Lab 6, Winter 2020

Introduction

In this lab, you will use R to use the bootstrap to compute standard errors for CER model estimates of five Northwest stocks in the **IntroCompFin** package: Amazon (amzn), Boeing (ba), Costco (cost), Nordstrom (jwn), and Starbucks (sbux). You will get started on the class project (see the Final Project under Assignments on Canvas for more details on the class project). This notebook walks you through all of the computations for the bootstrap part of the lab. You will use the following R packages

- **boot**
- **IntroCompFinR**
- **PerformanceAnalytics package**.
- **zoo**
- **xts**

Make sure to install these packages before you load them into R. As in the previous labs, use this notebook to answer all questions. Insert R chunks where needed. I will provide code hints below.

Reading

- Zivot, chapters 6 (CER Model), 7 (CER Model Estimation) and 8 (bootstrap)
- Ruppert and Matteson, chapter 6 (resampling) and Appendix sections 10, 11, 16 and 17

Load packages and set options

```
suppressPackageStartupMessages(library(IntroCompFinR))
suppressPackageStartupMessages(library(corrplot))
suppressPackageStartupMessages(library(PerformanceAnalytics))
suppressPackageStartupMessages(library(xts))
suppressPackageStartupMessages(library(boot))
options(digits = 3)
Sys.setenv(TZ="UTC")
```

Loading data and computing returns

Load the daily price data from **IntroCompFinR**, and create monthly returns over the period Jan 1998 through Dec 2014:

```
data(amznDailyPrices, baDailyPrices, costDailyPrices, jwnDailyPrices, sbuxDailyPrices)
fiveStocks = merge(amznDailyPrices, baDailyPrices, costDailyPrices, jwnDailyPrices, sbuxDailyPri
ces)
fiveStocks = to.monthly(fiveStocks, OHLC=FALSE)
```

```
## Warning in to.period(x, "months", indexAt = indexAt, name = name, ...): missing
## values removed from data
```

Next, let's compute monthly continuously compounded returns using the **PerformanceAnalytics** function `Return.Calculate()`

```
fiveStocksRet = na.omit(Return.calculate(fiveStocks, method = "log"))
head(fiveStocksRet, n=3)
```

```
##            AMZN      BA   COST    JWN   SBUX
## Feb 1998 0.2661  0.1332 0.1193 0.1220 0.0793
## Mar 1998 0.1049 -0.0401 0.0880 0.1062 0.1343
## Apr 1998 0.0704 -0.0404 0.0458 0.0259 0.0610
```

We removed the missing January return using the function `na.omit()` .

Part I: CER Model Estimation (repeat of lab 5)

Consider the CER Model for cc returns

$$\begin{aligned} R_{it} &= \mu_i + \epsilon_{it}, t = 1, \cdots, T \\ \epsilon_{it} &\sim \text{iid } N(0, \sigma_i^2) \\ \text{cov}(R_{it}, R_{jt}) &= \sigma_{i,j} \\ \text{cov}(R_{it}, R_{js}) &= 0 \text{ for } s \neq t \end{aligned}$$

where R_{it} denotes the cc return on asset i ($i = \text{AMZN}, \cdots, \text{SBUX}$).

1. Using sample descriptive statistics, give estimates for the model parameters $\mu_i, \sigma_i^2, \sigma_i, \sigma_{i,j}, \rho_{i,j}$.

```
muhat = apply(fiveStocksRet, 2, mean)
muhat
```

```
##    AMZN      BA    COST     JWN    SBUX
## 0.02042 0.00657 0.01017 0.01049 0.01458
```

```
sigma2hat = apply(fiveStocksRet, 2, var)
sigma2hat
```

```
##    AMZN      BA    COST     JWN    SBUX
## 0.02814 0.00764 0.00584 0.01298 0.01111
```

```
sigmahat = apply(fiveStocksRet, 2, sd)
sigmahat
```

```
##   AMZN     BA   COST    JWN   SBUX
## 0.1678 0.0874 0.0764 0.1140 0.1054
```

```
covmat = var(fiveStocksRet)
covmat
```

```
##         AMZN      BA    COST     JWN    SBUX
## AMZN 0.02814 0.00208 0.00404 0.00714 0.00546
## BA   0.00208 0.00764 0.00118 0.00319 0.00237
## COST 0.00404 0.00118 0.00584 0.00355 0.00271
## JWN  0.00714 0.00319 0.00355 0.01298 0.00468
## SBUX 0.00546 0.00237 0.00271 0.00468 0.01111
```

```
cormat = cor(fiveStocksRet)
cormat
```

```
##       AMZN    BA  COST   JWN  SBUX
## AMZN 1.000 0.142 0.315 0.374 0.309
## BA   0.142 1.000 0.177 0.321 0.258
## COST 0.315 0.177 1.000 0.407 0.336
## JWN  0.374 0.321 0.407 1.000 0.390
## SBUX 0.309 0.258 0.336 0.390 1.000
```

```
covhat = covmat[lower.tri(covmat)]
rhohat = cormat[lower.tri(cormat)]
names(covhat) <- names(rhohat) <-
c("AMZN,BA","AMZN,COST","AMZN,JWN","AMZN,SBUX", "BA,COST", "BA,JWN","BA,SBUX", "COST,JWN","COST,
SBUX","JWN,SBUX")
covhat
```

```
##   AMZN,BA AMZN,COST  AMZN,JWN AMZN,SBUX   BA,COST    BA,JWN   BA,SBUX  COST,JWN
##   0.00208   0.00404   0.00714   0.00546   0.00118   0.00319   0.00237   0.00355
## COST,SBUX  JWN,SBUX
##   0.00271   0.00468
```

```
rhohat
```

```
##   AMZN,BA AMZN,COST  AMZN,JWN AMZN,SBUX   BA,COST    BA,JWN   BA,SBUX  COST,JWN
##     0.142     0.315     0.374     0.309     0.177     0.321     0.258     0.407
## COST,SBUX  JWN,SBUX
##     0.336     0.390
```

2. For each estimate of the above parameters (except $\sigma_{i,j}$) compute the estimated standard error. That is, compute $\widehat{\mathrm{SE}}(\hat{\mu}_i)$, $\widehat{\mathrm{SE}}(\hat{\sigma}_i^2)$, $\widehat{\mathrm{SE}}(\hat{\sigma}_i)$, and $\widehat{\mathrm{SE}}(\hat{\rho}_{ij})$. Briefly comment on the precision of the estimates. We will compare the bootstrap SE values to these values.

```
n.obs = nrow(fiveStocksRet)
seMuhat = sigmahat/sqrt(n.obs)
cbind(muhat, seMuhat)
```

```
##        muhat seMuhat
## AMZN 0.02042 0.01177
## BA   0.00657 0.00613
## COST 0.01017 0.00536
## JWN  0.01049 0.00800
## SBUX 0.01458 0.00740
```

The model best estimates the μ value for Costco wit Amazon's estimate being the least accurate.

```
seSigma2hat = sigma2hat/sqrt(n.obs/2)
seSigmahat = sigmahat/sqrt(2*n.obs)
cbind(sigma2hat, seSigma2hat, sigmahat, seSigmahat)
```

```
##      sigma2hat seSigma2hat sigmahat seSigmahat
## AMZN   0.02814    0.002793   0.1678    0.00833
## BA     0.00764    0.000758   0.0874    0.00434
## COST   0.00584    0.000580   0.0764    0.00379
## JWN    0.01298    0.001289   0.1140    0.00566
## SBUX   0.01111    0.001102   0.1054    0.00523
```

These SE values indicate that these measures for σ^2 and σ are pretty good, all being smaller than the SE values for μ.

```
seRhohat = (1-rhohat^2)/sqrt(n.obs)
cbind(rhohat, seRhohat)
```

```
##           rhohat seRhohat
## AMZN,BA    0.142   0.0688
## AMZN,COST  0.315   0.0632
## AMZN,JWN   0.374   0.0604
## AMZN,SBUX  0.309   0.0635
## BA,COST    0.177   0.0680
## BA,JWN     0.321   0.0630
## BA,SBUX    0.258   0.0655
## COST,JWN   0.407   0.0585
## COST,SBUX  0.336   0.0622
## JWN,SBUX   0.390   0.0595
```

The $\widehat{\mathrm{SE}}(\hat{\rho}_{ij})$ value is small compared to the $\hat{\rho}_{ij}$ values meaning that they are adequate estimates with the higher correlations having smaller $\widehat{\mathrm{SE}}(\hat{\rho}_{ij})$ values.

Part II: Bootstrapping the CER Model Estimates

1. For each estimate of the above parameters (except $\sigma_{i,j}$) compute the estimated standard error using the bootstrap with 999 bootstrap replications. That is, compute $\widehat{\mathrm{SE}}_{boot}(\hat{\mu}_i)$, $\widehat{\mathrm{SE}}_{boot}(\hat{\sigma}_i^2)$, $\widehat{\mathrm{SE}}_{boot}(\hat{\sigma}_i)$, and $\widehat{\mathrm{SE}}_{boot}(\hat{\rho}_{ij})$. Compare the bootstrap standard errors to the analytic standard errors you computed above.

```
amznRet = fiveStocksRet[,1]
baRet = fiveStocksRet[,2]
costRet = fiveStocksRet[,3]
jwnRet = fiveStocksRet[,4]
sbuxRet = fiveStocksRet[,5]
```

```
bootMean <- function(x,idx) {
  ans = mean(x[idx])
}
set.seed(123)
amznBootMean = boot(amznRet, bootMean, 999)
baBootMean = boot(baRet, bootMean, 999)
costBootMean = boot(costRet, bootMean, 999)
jwnBootMean = boot(jwnRet, bootMean, 999)
sbuxBootMean = boot(sbuxRet, bootMean, 999)
amznBootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = amznRet, statistic = bootMean, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*   0.0204 0.000505      0.0115
```

```
baBootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = baRet, statistic = bootMean, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*  0.00657 -0.000227     0.00628
```

```
costBootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = costRet, statistic = bootMean, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.0102 -0.000115      0.0054
```

```
jwnBootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = jwnRet, statistic = bootMean, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*   0.0105 0.000235     0.00783
```

```
sbuxBootMean
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = sbuxRet, statistic = bootMean, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*   0.0146 0.000174     0.00748
```

The bootstrap SE and the analytical SE are relatively the same as eachother with minimal differences.

```
bootVar <- function(x,idx) {
  ans = var(x[idx])
}

amznBootVar = boot(amznRet, bootVar, 999)
baBootVar = boot(baRet, bootVar, 999)
costBootVar = boot(costRet, bootVar, 999)
jwnBootVar = boot(jwnRet, bootVar, 999)
sbuxBootVar = boot(sbuxRet, bootVar, 999)
amznBootVar
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = amznRet, statistic = bootVar, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.0281 -6.03e-05     0.00449
```

```
baBootVar
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = baRet, statistic = bootVar, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*  0.00764 -1.97e-05     0.00108
```

```
costBootVar
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = costRet, statistic = bootVar, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*  0.00584 -4.4e-05     0.00144
```

```
jwnBootVar
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = jwnRet, statistic = bootVar, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.013 -0.000103     0.00196
```

```
sbuxBootVar
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = sbuxRet, statistic = bootVar, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.0111 -3.06e-06     0.00191
```

For the variance, the SE values for boot are all higher than the analytical SE’s for the assets.

```
bootSD <- function(x,idx) {
  ans = sd(x[idx])
}

amznBootSD = boot(amznRet, bootSD, 999)
baBootSD = boot(baRet, bootSD, 999)
costBootSD = boot(costRet, bootSD, 999)
jwnBootSD = boot(jwnRet, bootSD, 999)
sbuxBootSD = boot(sbuxRet, bootSD, 999)
amznBootSD
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = amznRet, statistic = bootSD, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.168 -0.000461      0.0133
```

```
baBootSD
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = baRet, statistic = bootSD, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.0874 -0.000408     0.00666
```

```
costBootSD
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = costRet, statistic = bootSD, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.0764 -0.000661     0.00933
```

```
jwnBootSD
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = jwnRet, statistic = bootSD, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.114 -0.000826     0.00894
```

```
sbuxBootSD
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = sbuxRet, statistic = bootSD, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.105 -0.000237     0.00915
```

Again the values for the bootstrap SE are higher than the analytical SE values for standard deviation.

```r
return.mat <- as.matrix(fiveStocksRet)
bootRho <- function(x.mat,idx) {
  ans = cor(x.mat[idx,])[1,2]
}

amzn.baBootRho = boot(return.mat[,c(1,2)], bootRho, 999)
amzn.costBootRho = boot(return.mat[,c(1,3)], bootRho, 999)
amzn.jwnBootRho = boot(return.mat[,c(1,4)], bootRho, 999)
amzn.sbuxBootRho = boot(return.mat[,c(1,5)], bootRho, 999)
ba.costBootRho = boot(return.mat[,c(2,3)], bootRho, 999)
ba.jwnBootRho = boot(return.mat[,c(2,4)], bootRho, 999)
ba.sbuxBootRho = boot(return.mat[,c(2,5)], bootRho, 999)
cost.jwnBootRho = boot(return.mat[,c(3,4)], bootRho, 999)
cost.sbuxBootRho = boot(return.mat[,c(3,5)], bootRho, 999)
jwn.sbuxBootRho = boot(return.mat[,c(4,5)], bootRho, 999)
amzn.baBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(1, 2)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    0.142 -0.00364      0.0975
```

```r
amzn.costBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(1, 3)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    0.315 0.00443       0.082
```

```r
amzn.jwnBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(1, 4)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.374 -0.00286      0.0687
```

```
amzn.sbuxBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(1, 5)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    0.309 0.00323       0.064
```

```
ba.costBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(2, 3)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*    0.177 -0.000498      0.0752
```

```
ba.jwnBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(2, 4)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    0.321 -0.00441      0.0838
```

```
ba.sbuxBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(2, 5)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    0.258 -0.00181      0.0783
```

```
cost.jwnBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(3, 4)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    0.407 0.00612      0.0558
```

```
cost.sbuxBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(3, 5)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    0.336  0.0106      0.0957
```

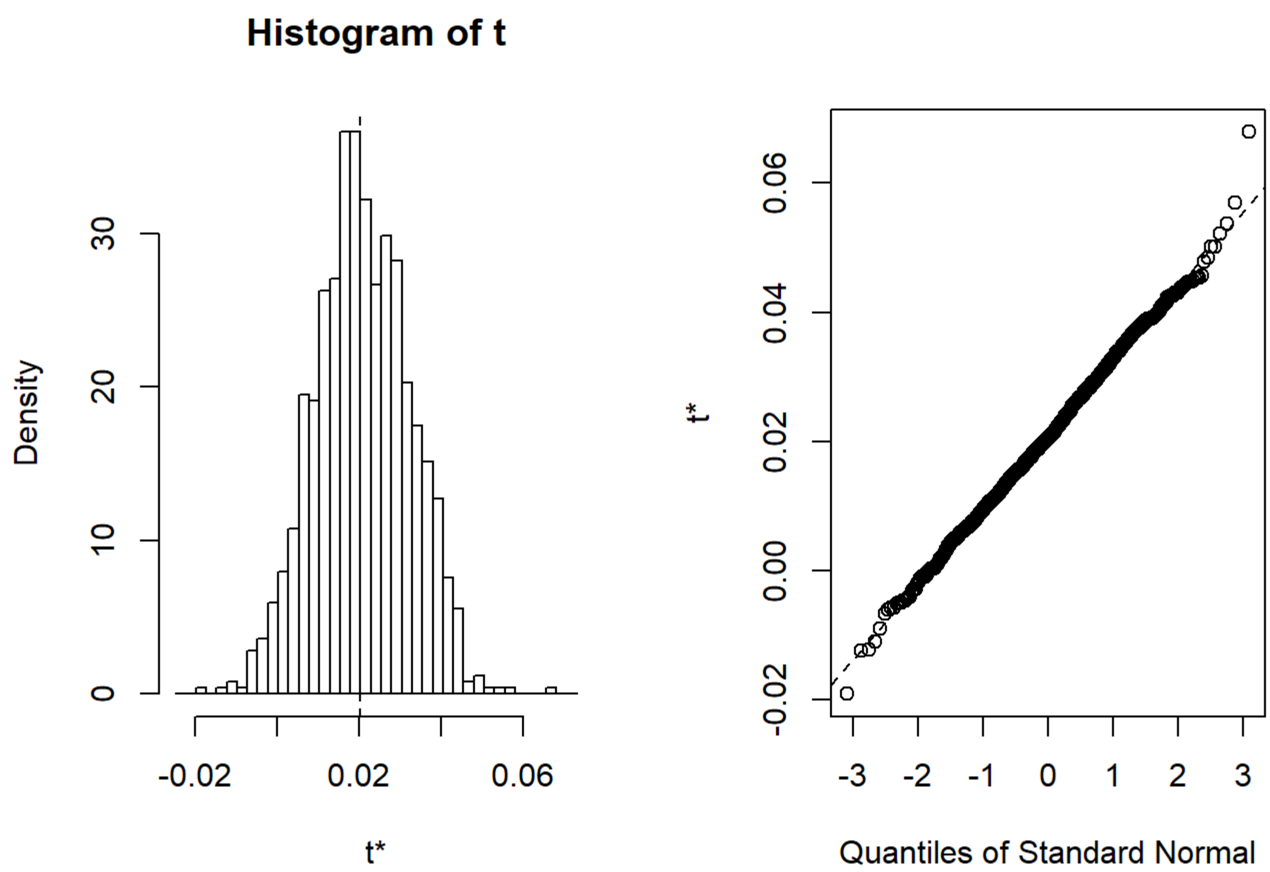
```
jwn.sbuxBootRho
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = return.mat[, c(4, 5)], statistic = bootRho, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*     0.39 -0.000948      0.0899
```

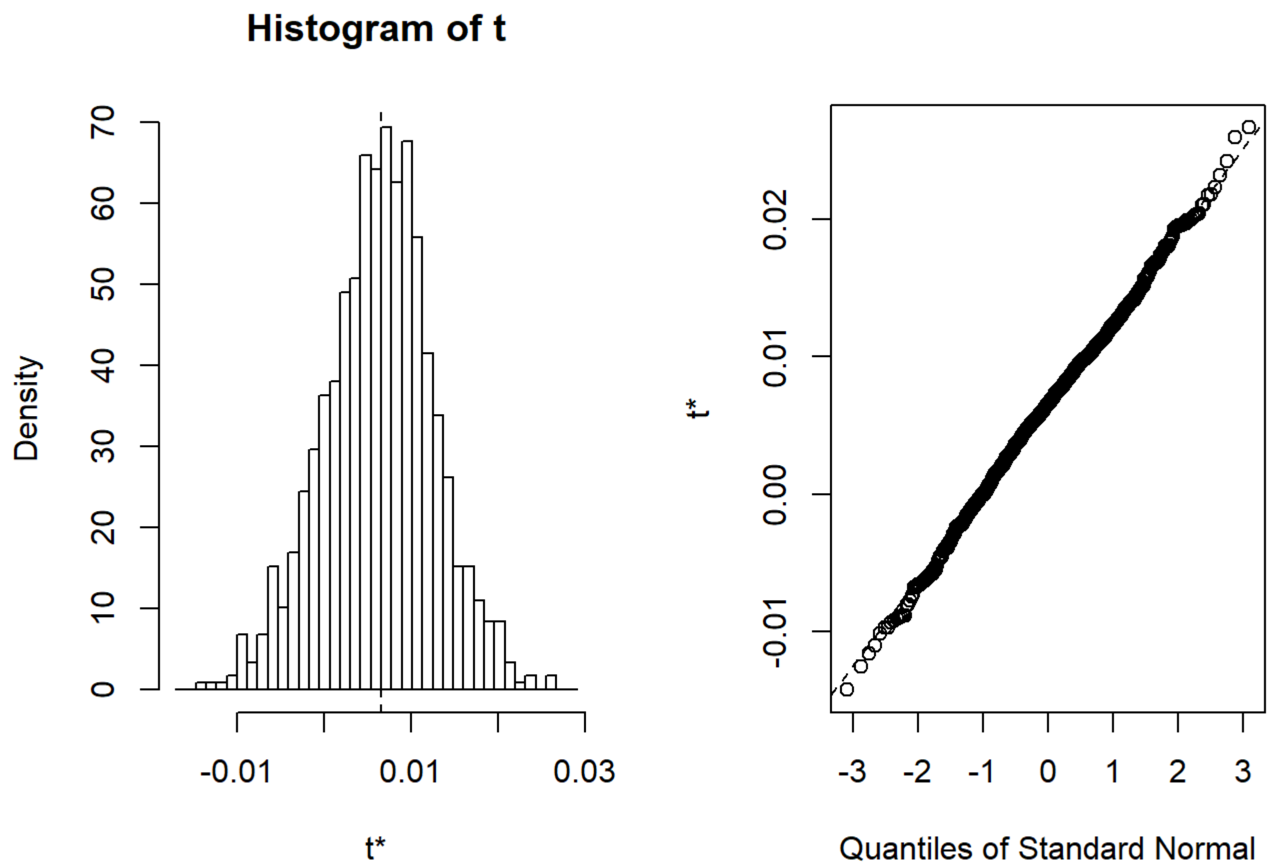
The bootstra SE values are all higher than the analytical SE values, besides for Costco and Nordstrom.

2. Plot the histogram and qq-plot of the bootstrap distributions you computed from the previous question. Do the bootstrap distributions look normal?

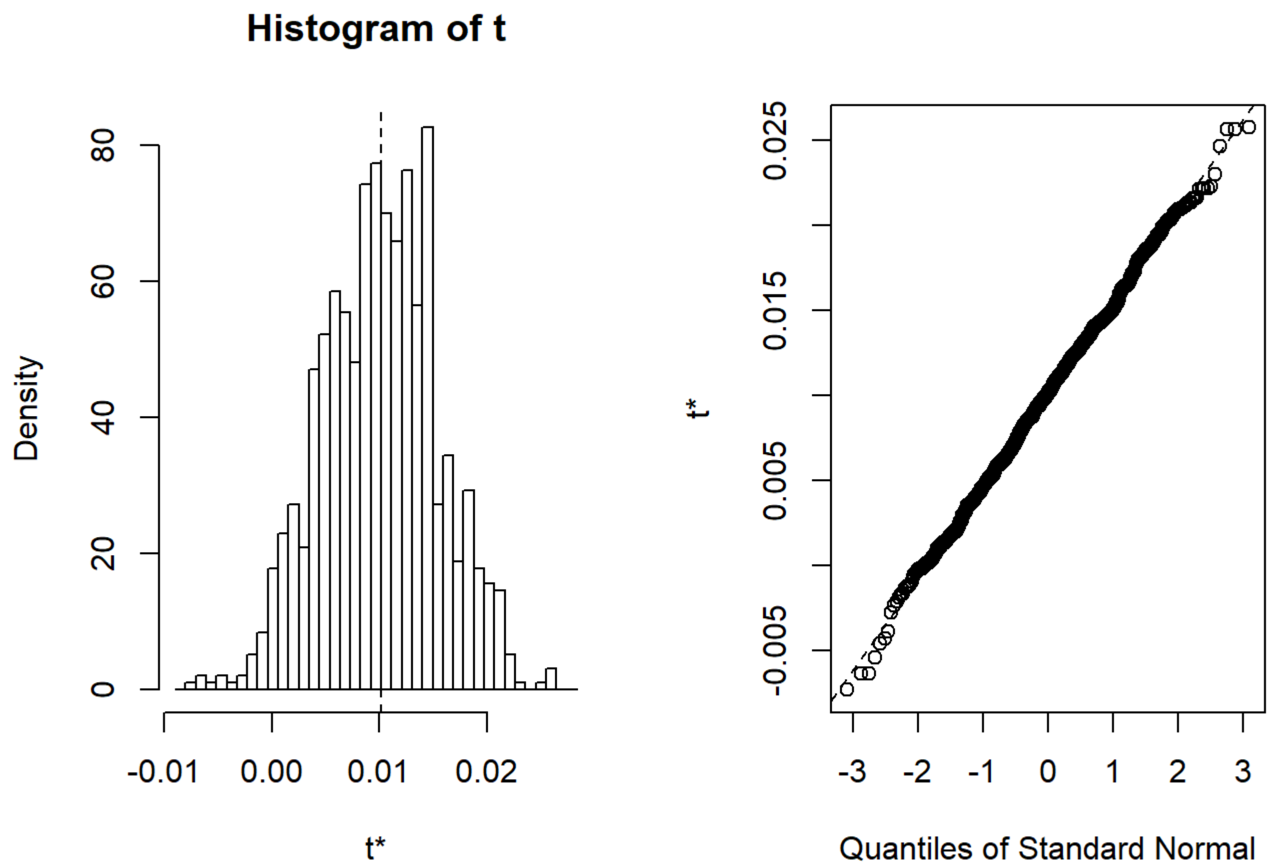
```
plot(amznBootMean)
```



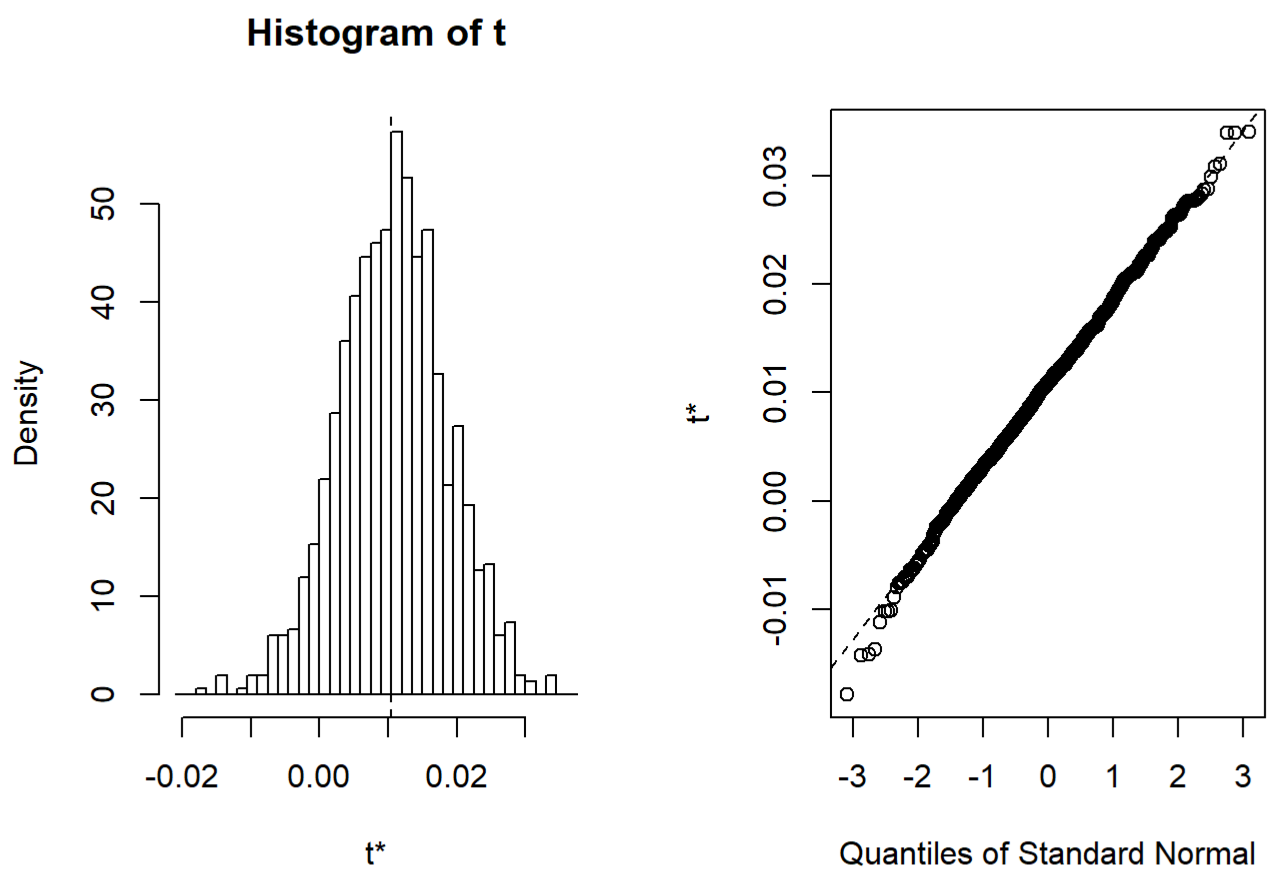

```
plot(baBootMean)
```



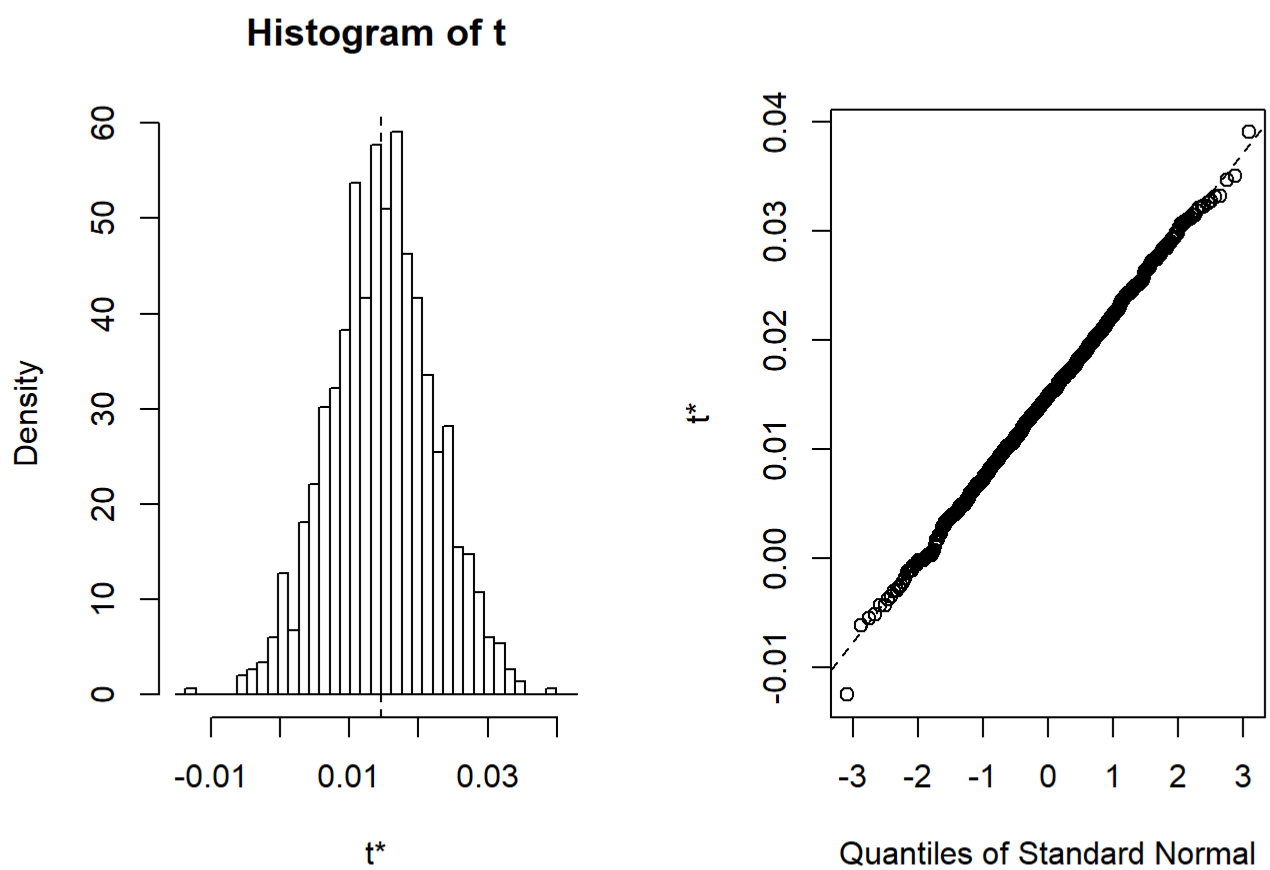

```
plot(costBootMean)
```



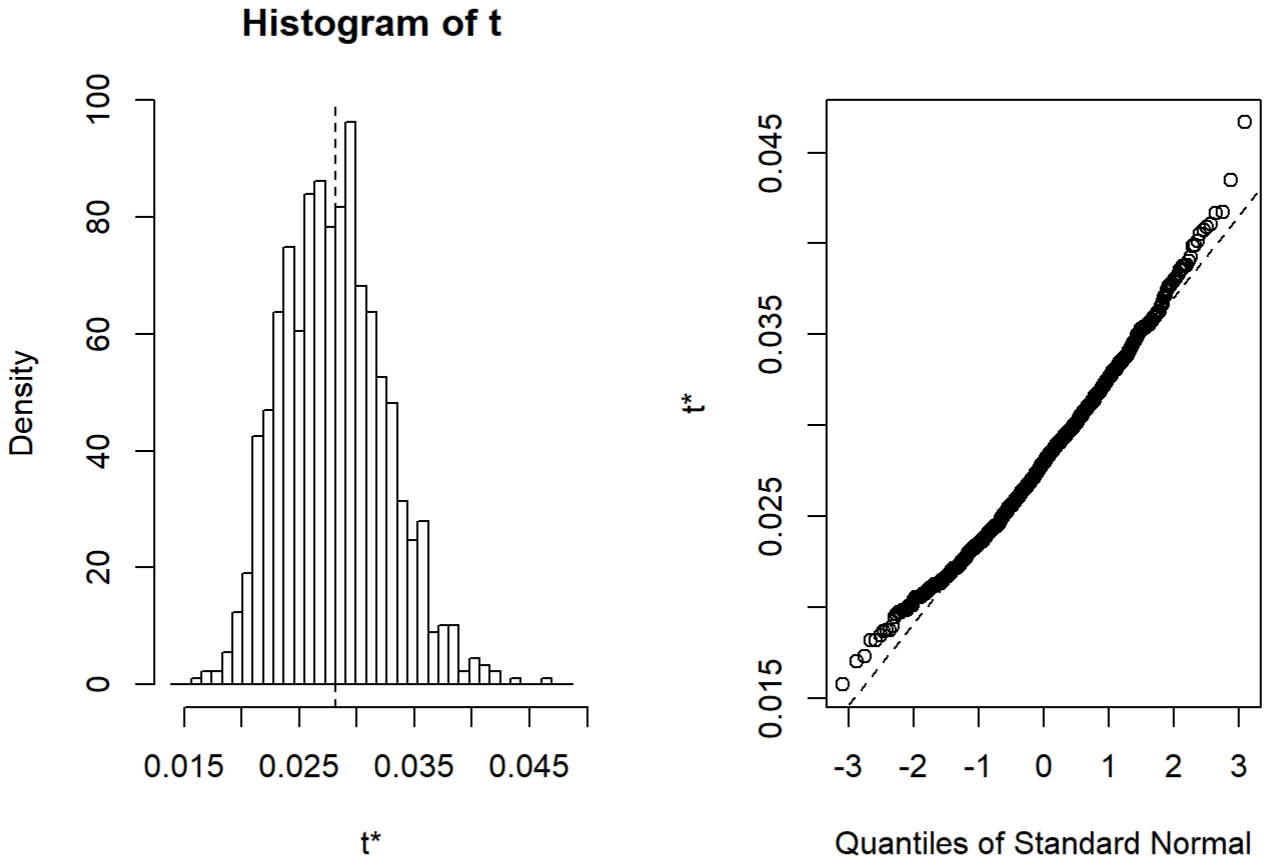

```
plot(jwnBootMean)
```



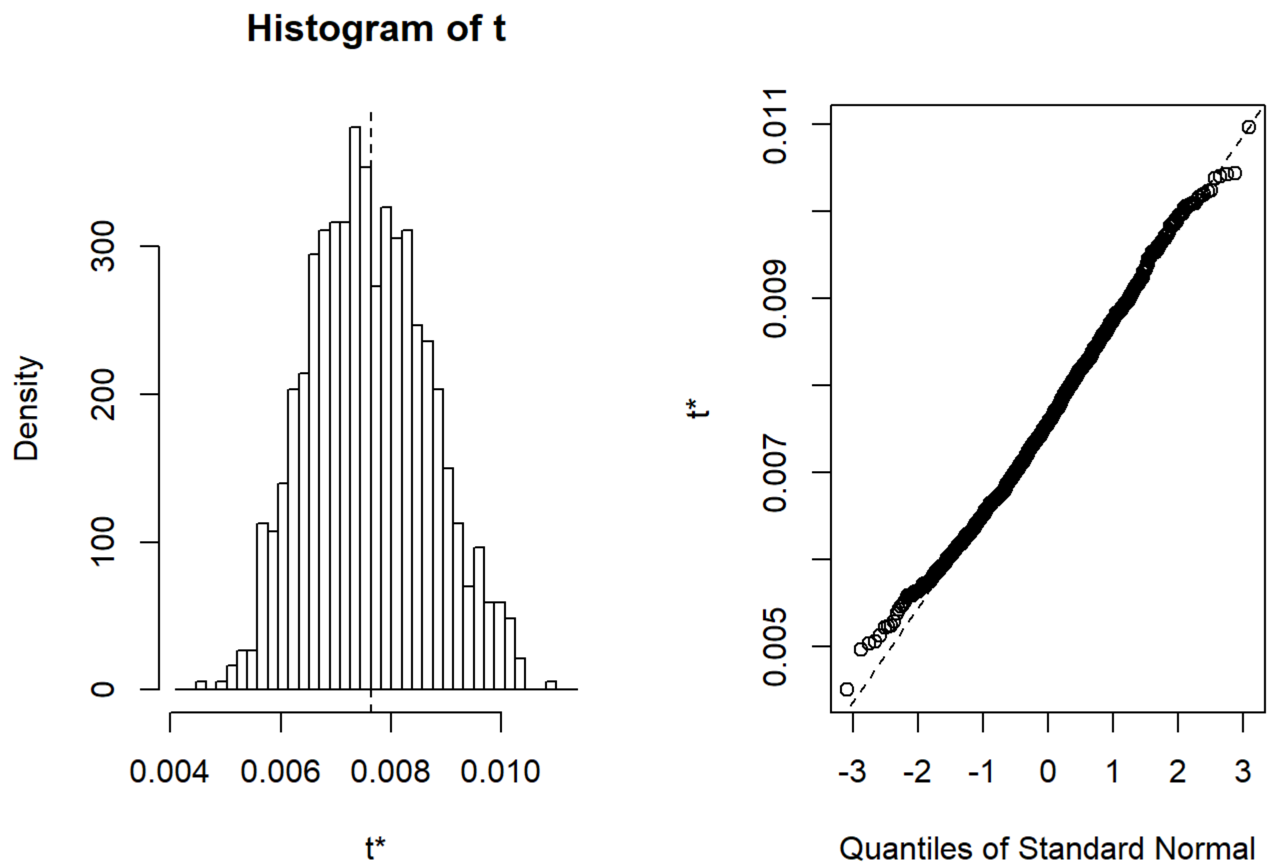

```
plot(sbuxBootMean)
```



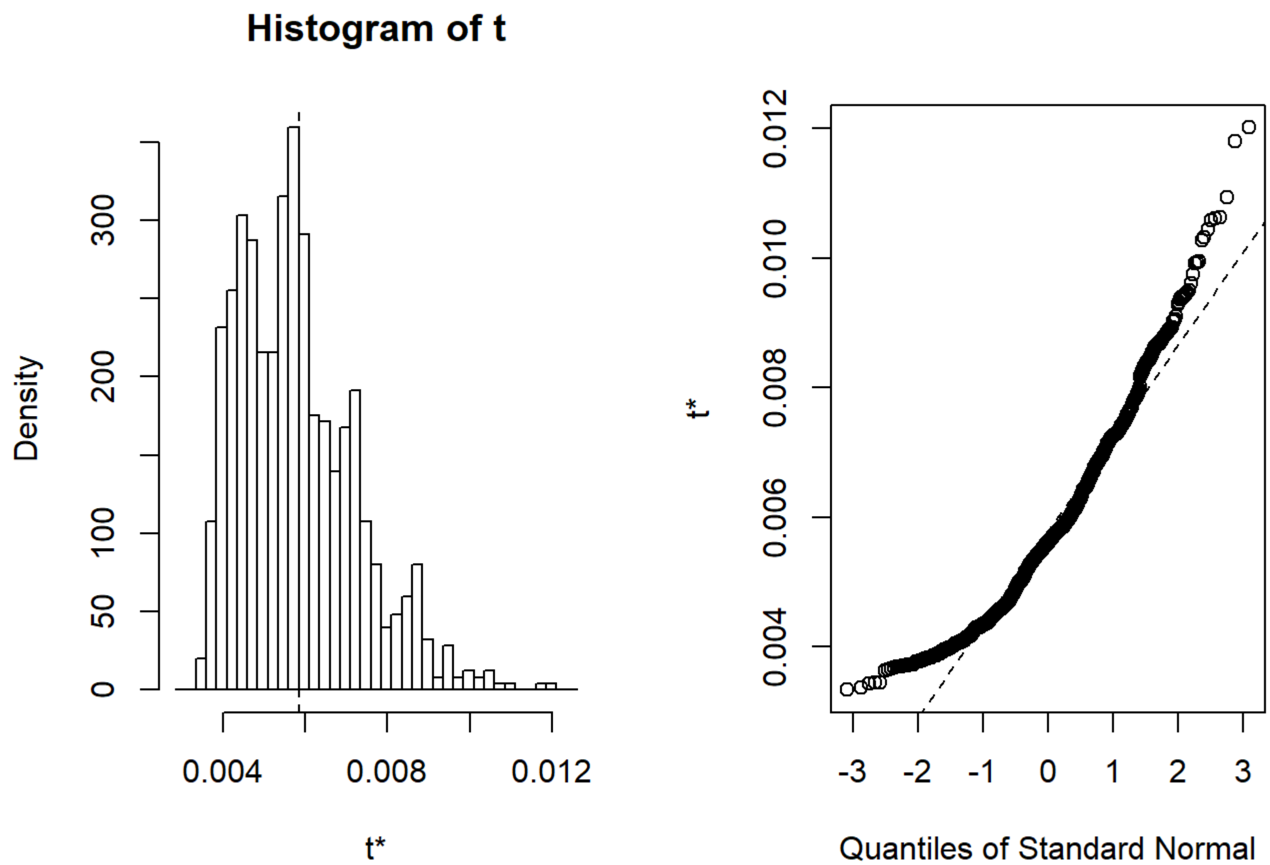

```
plot(amznBootVar)
```



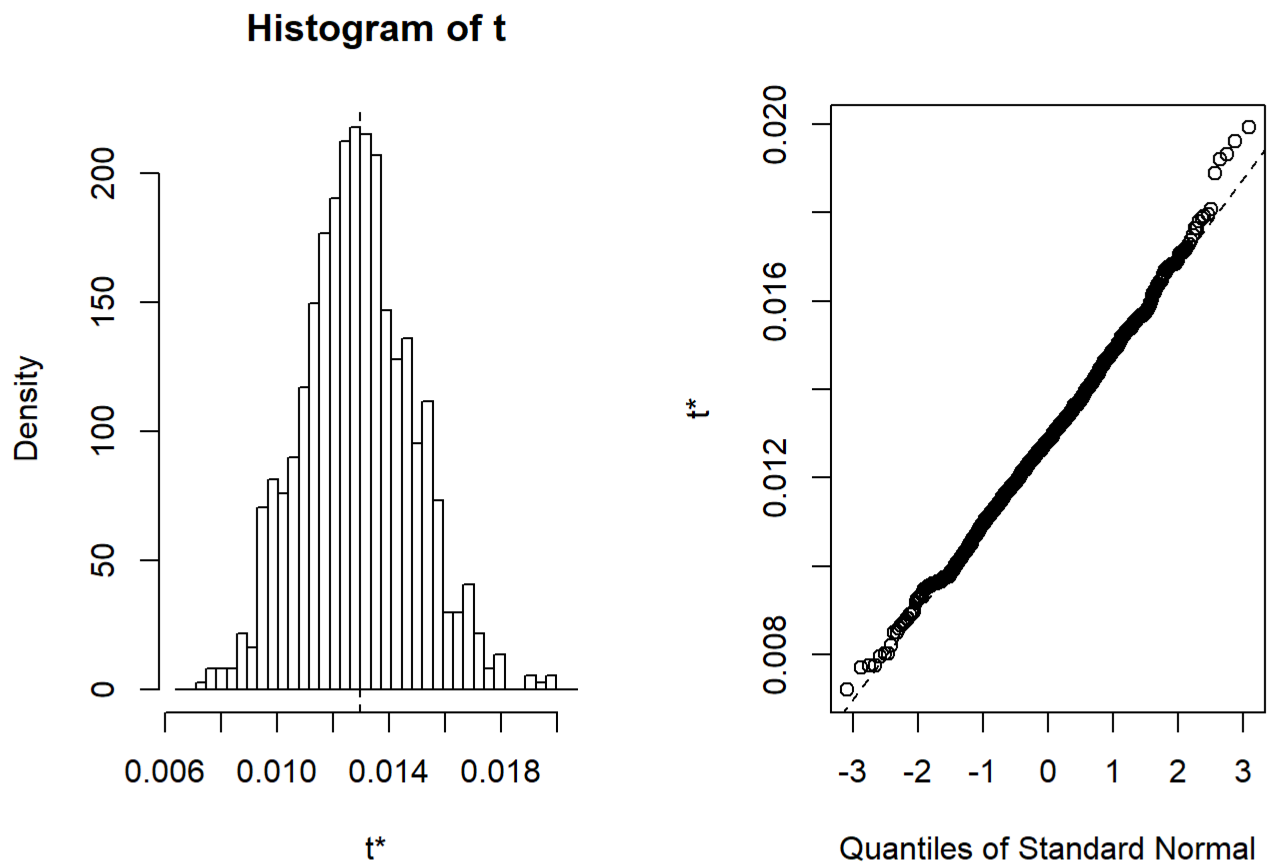

```
plot(baBootVar)
```



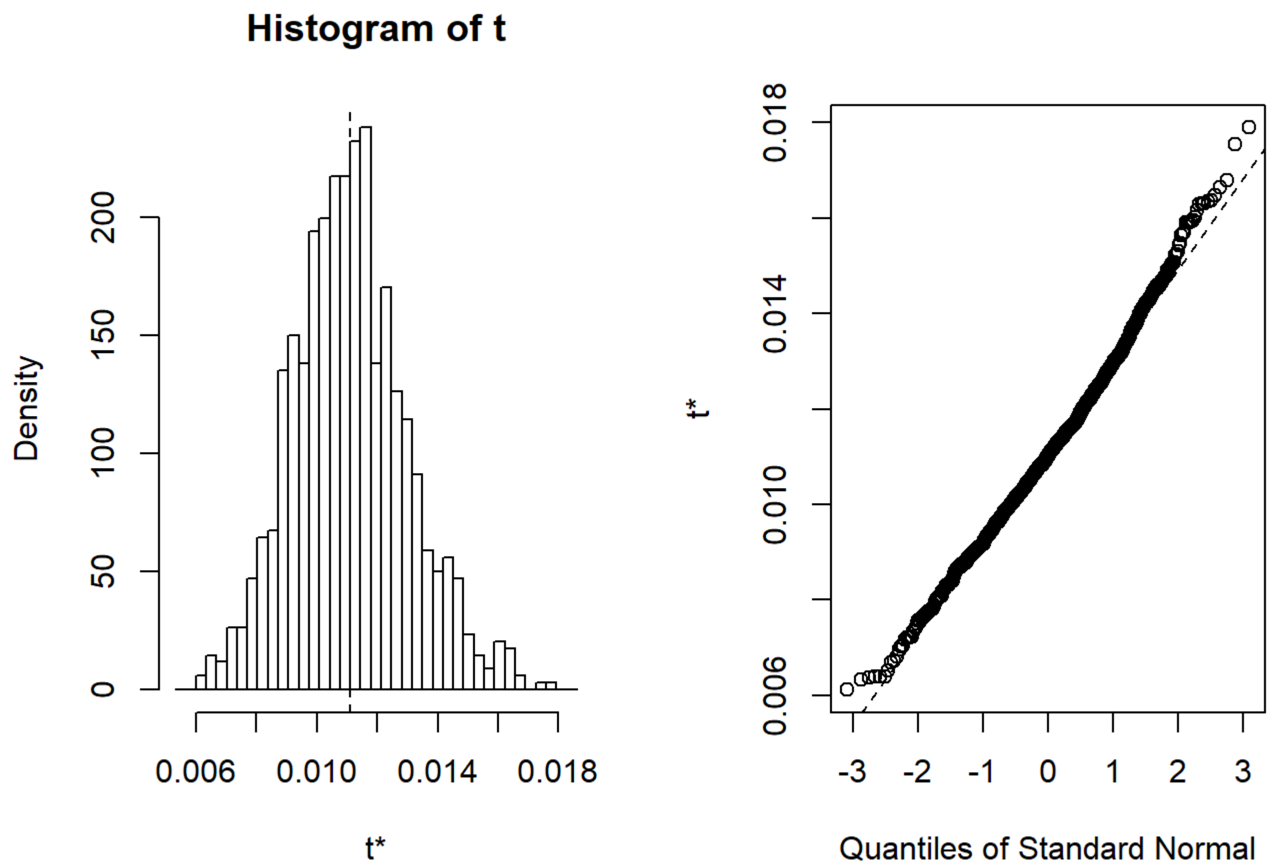

```
plot(costBootVar)
```



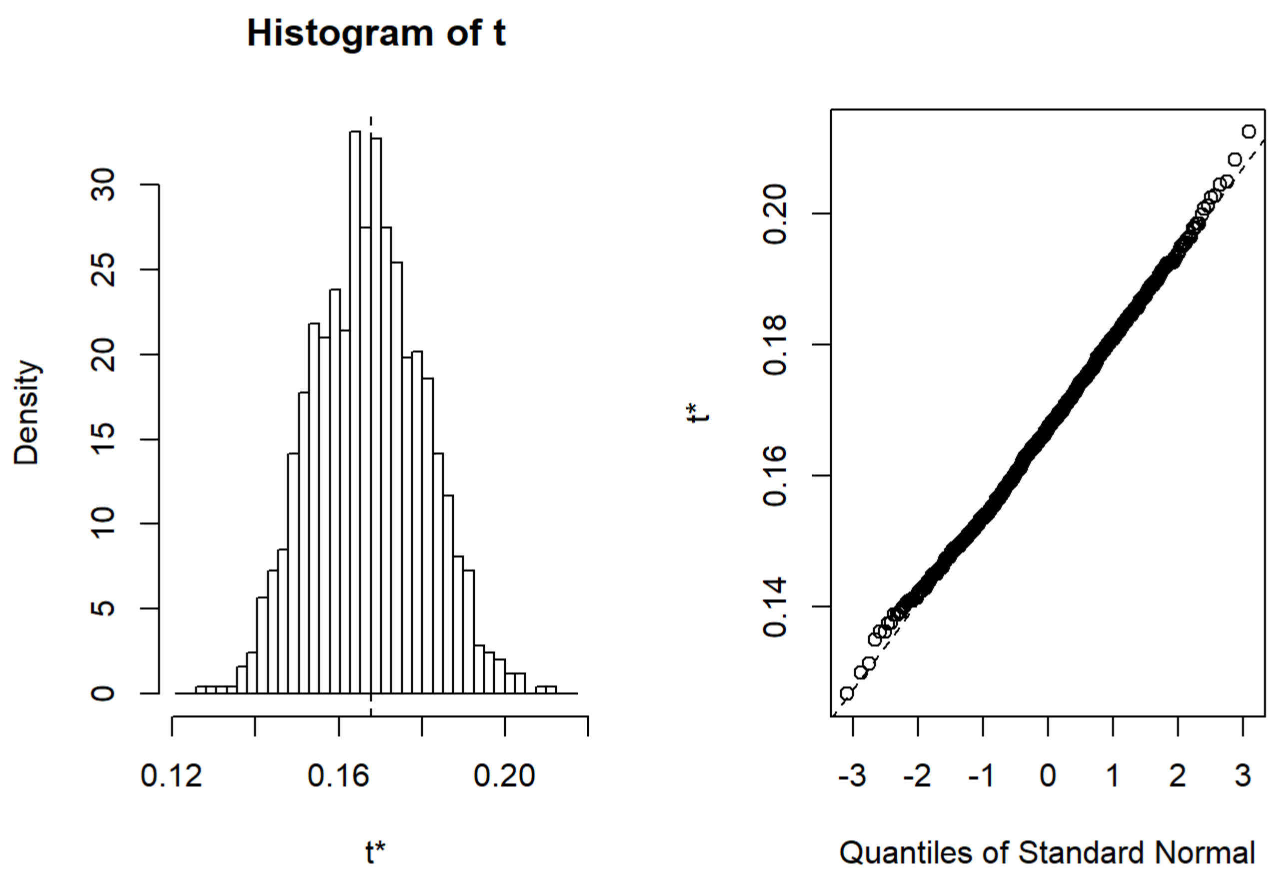

```
plot(jwnBootVar)
```



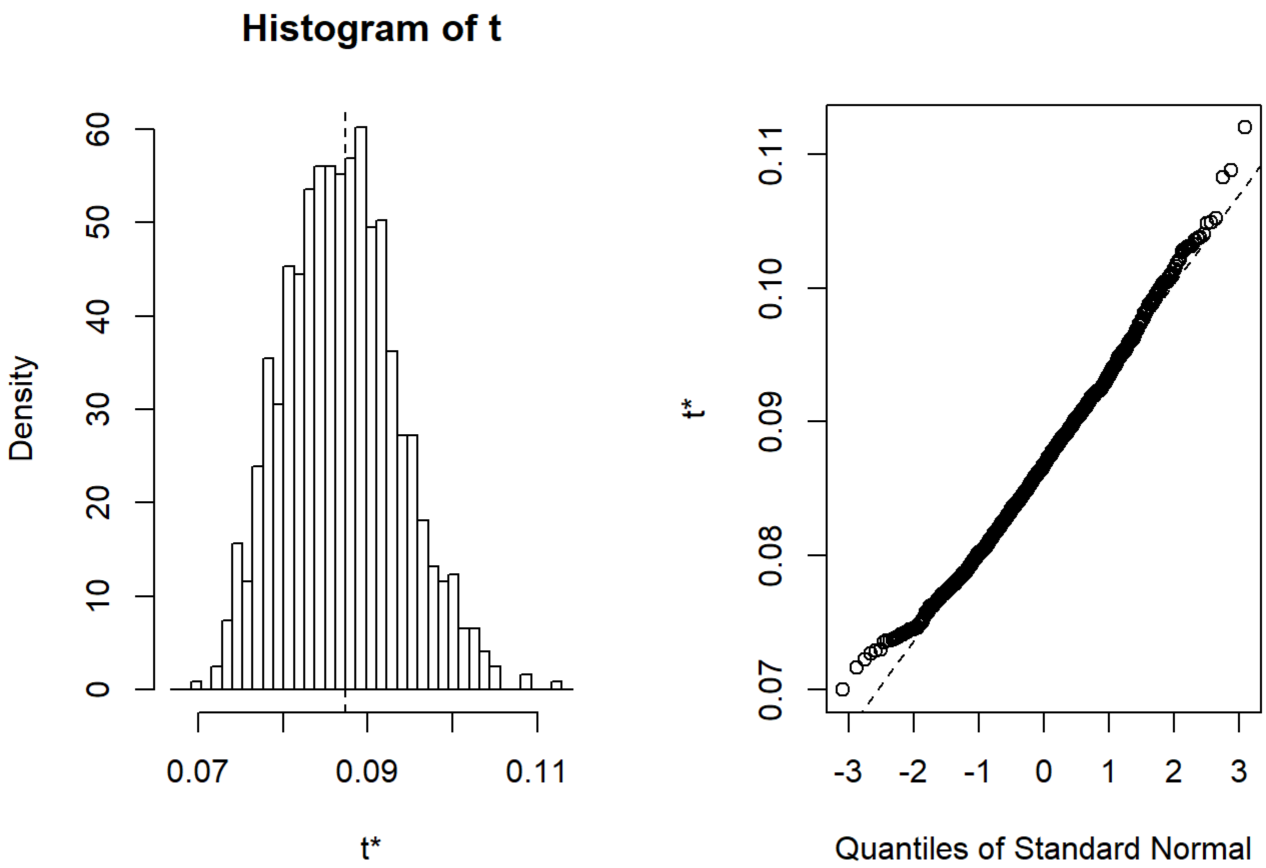

```
plot(sbuxBootVar)
```



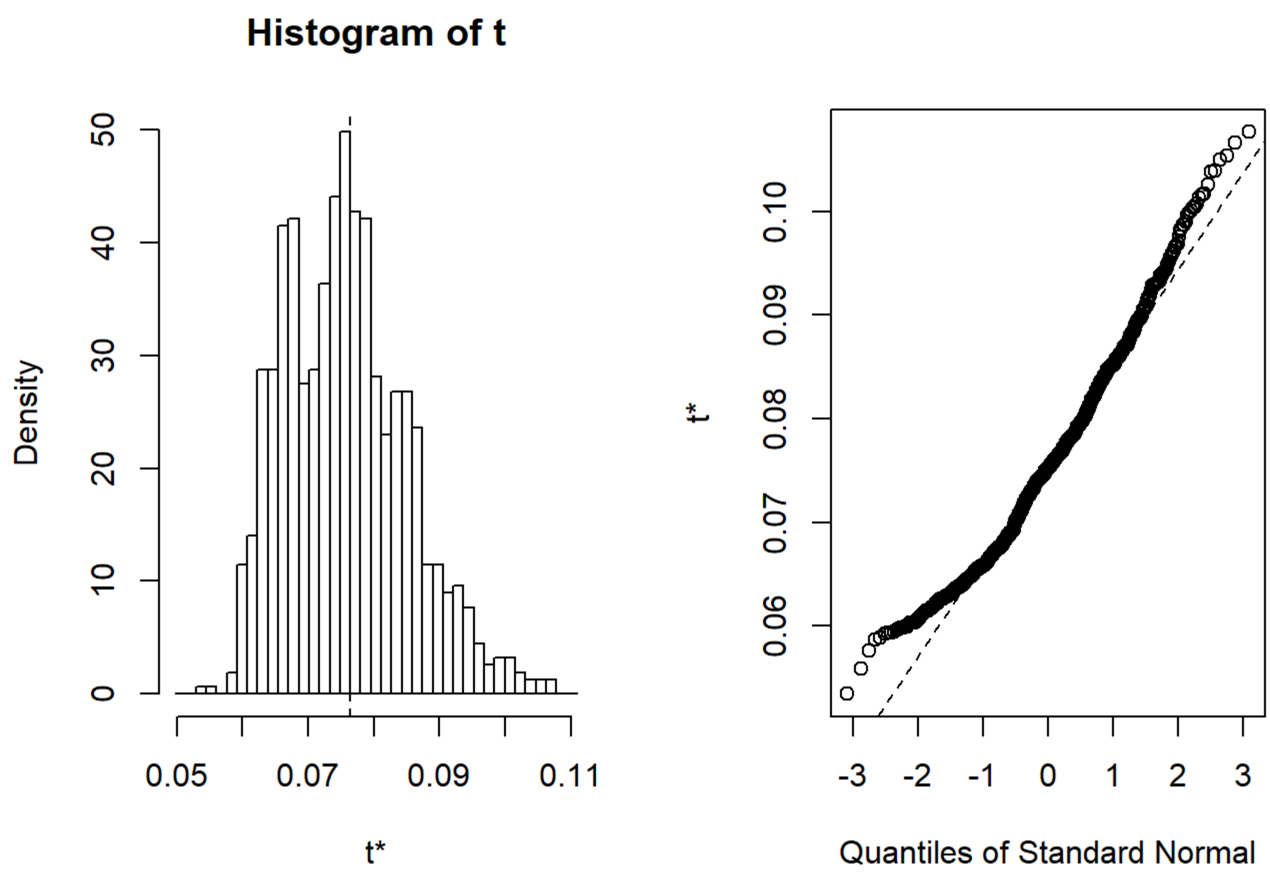

```
plot(amznBootSD)
```



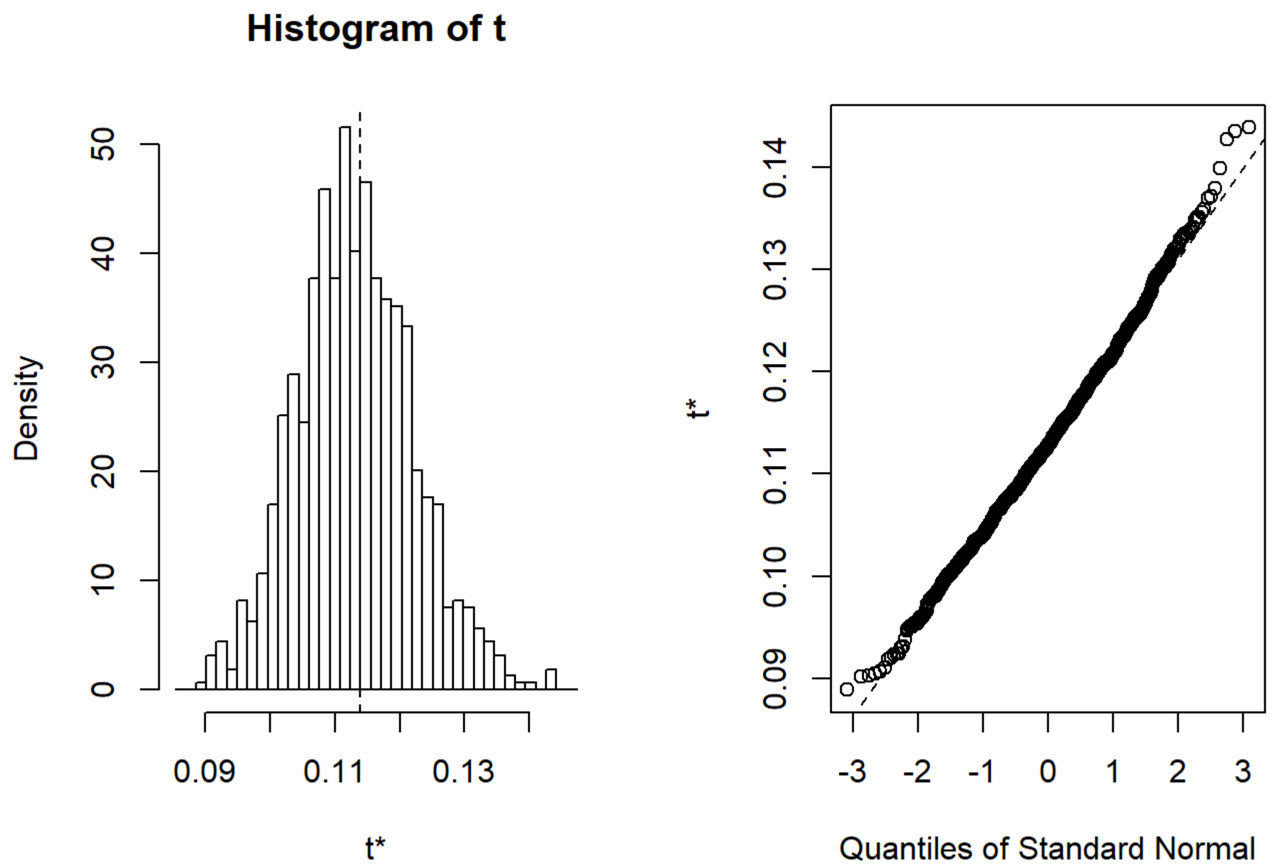

```
plot(baBootSD)
```



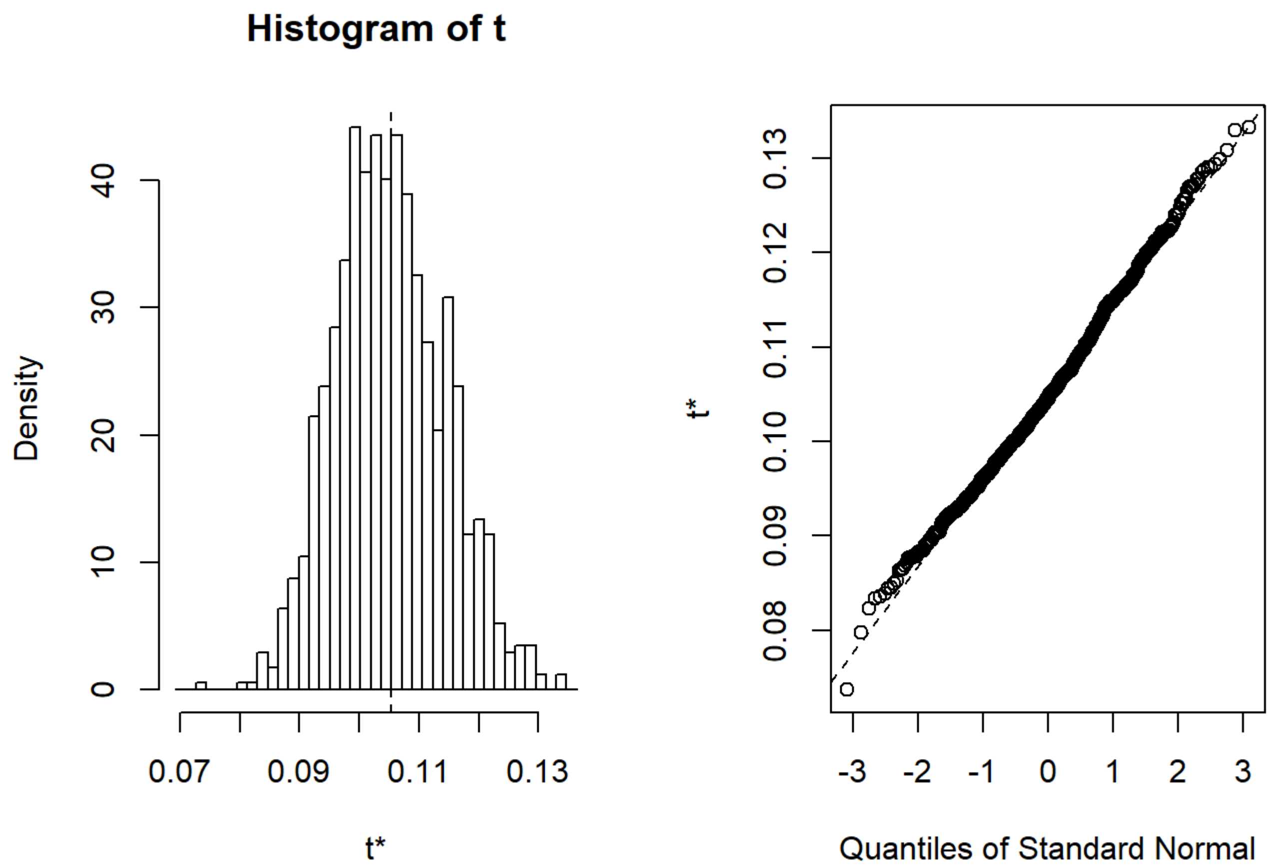

```
plot(costBootSD)
```



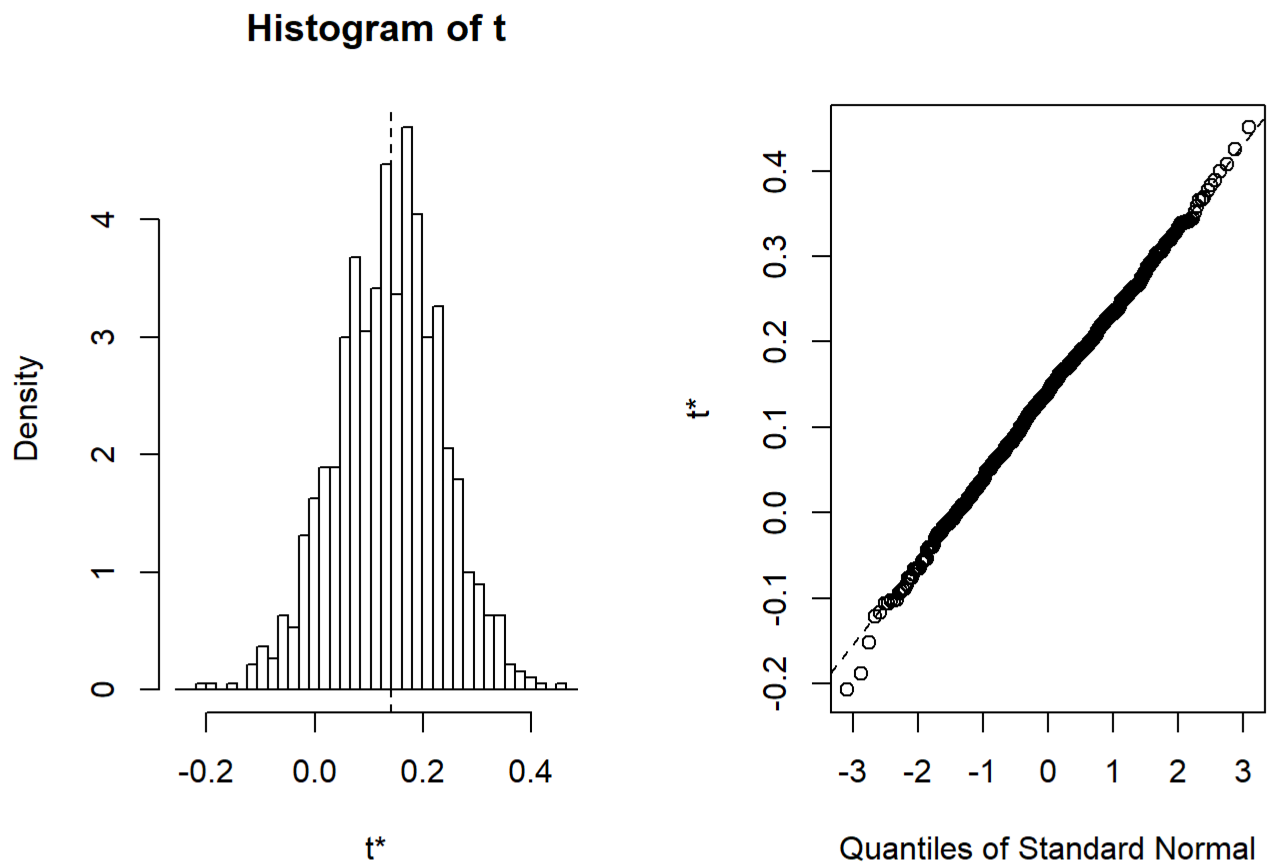

```
plot(jwnBootSD)
```



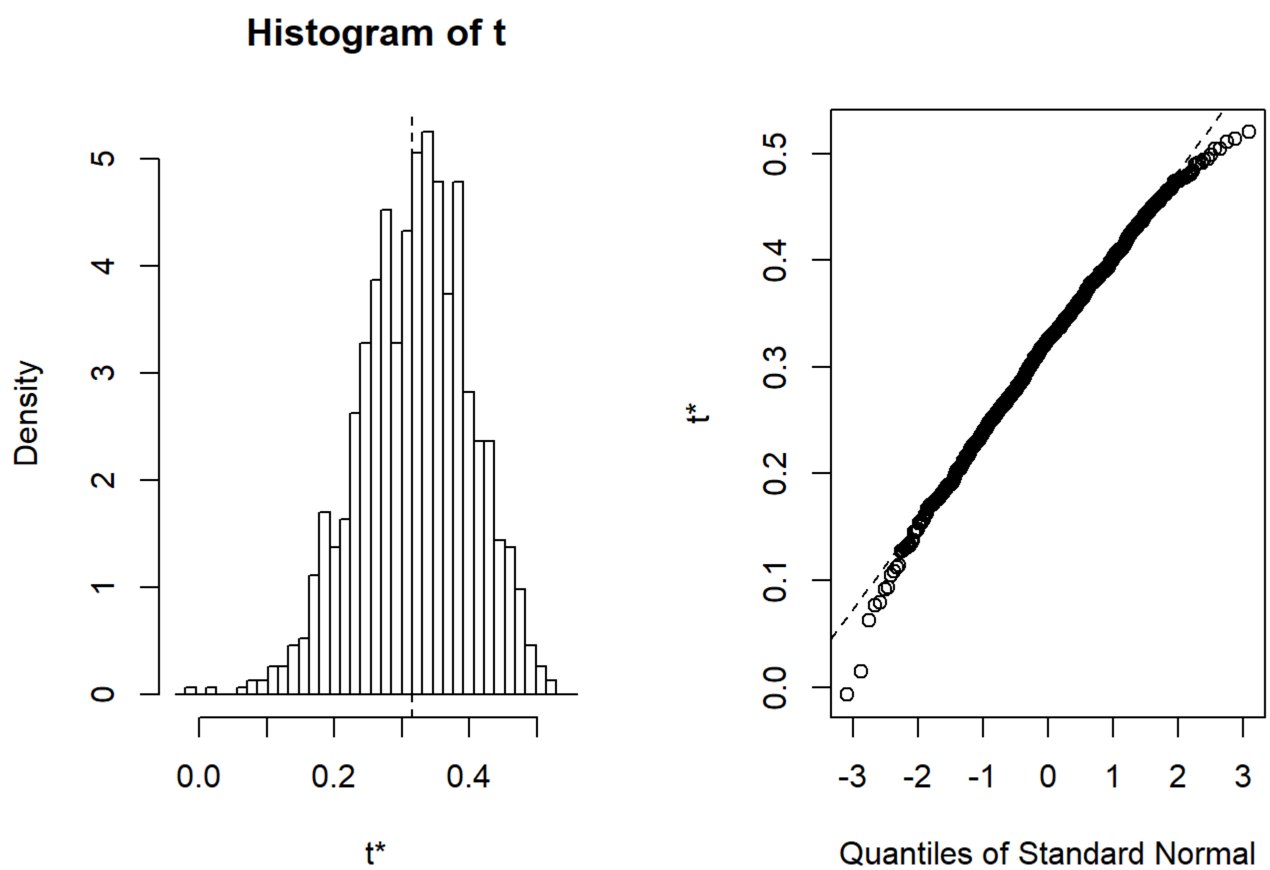

```
plot(sbuxBootSD)
```



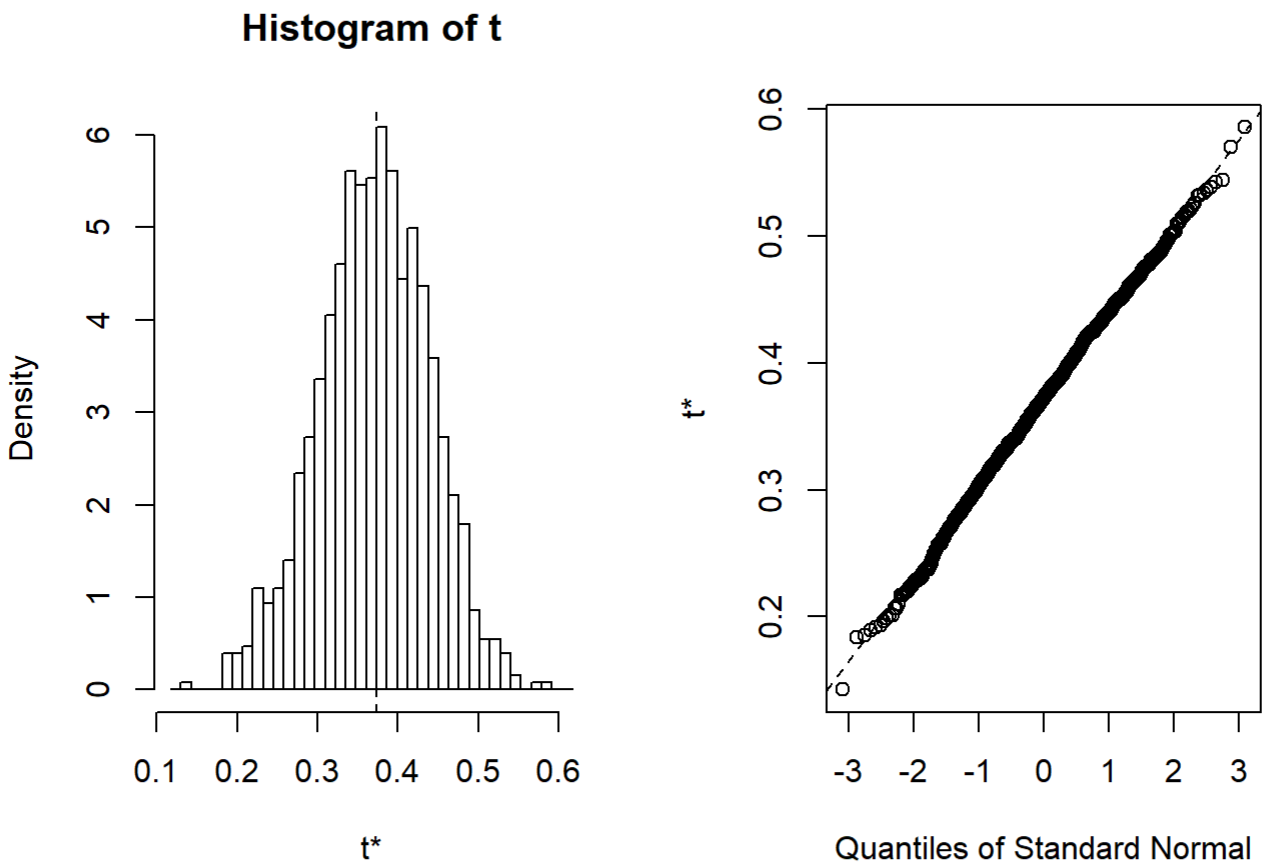

```
plot(amzn.baBootRho)
```



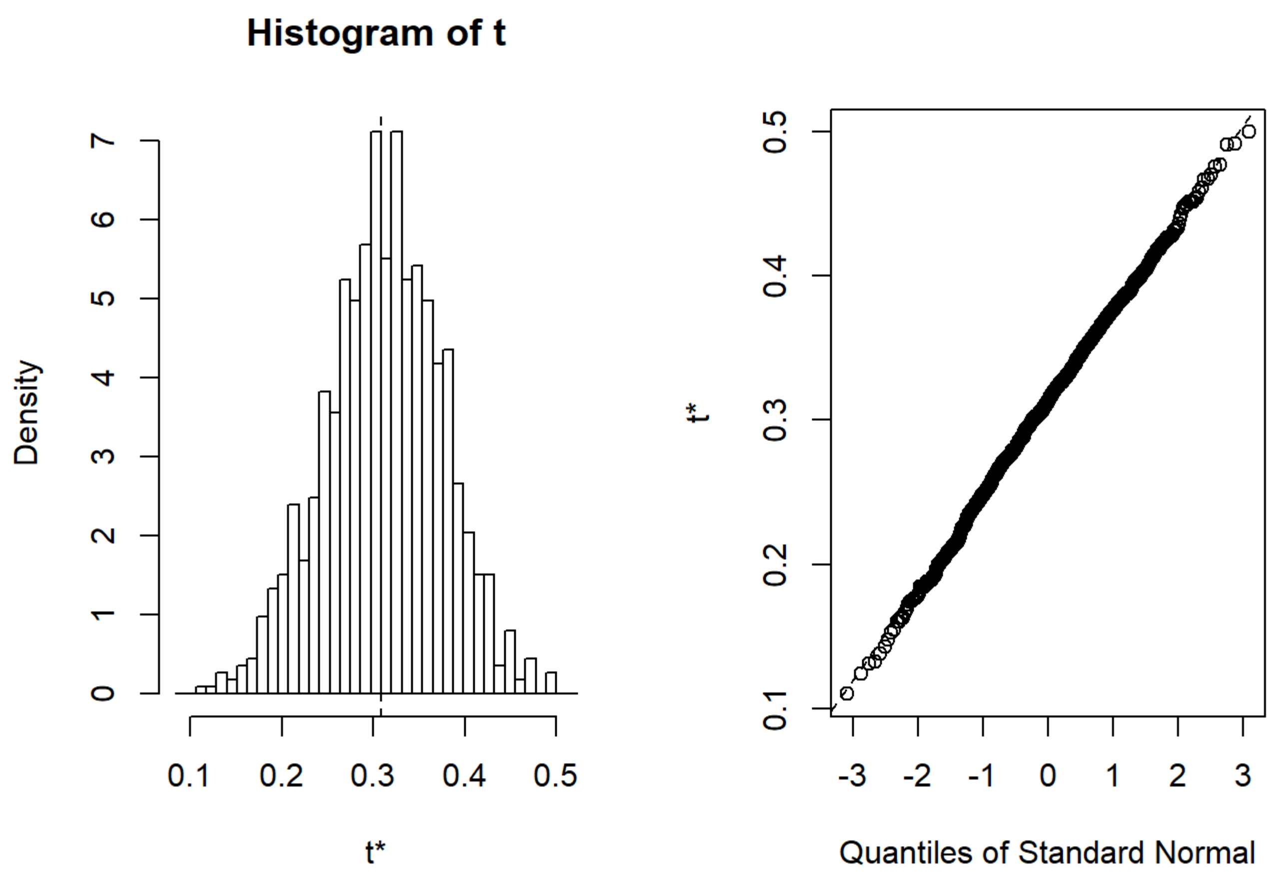

```
plot(amzn.costBootRho)
```



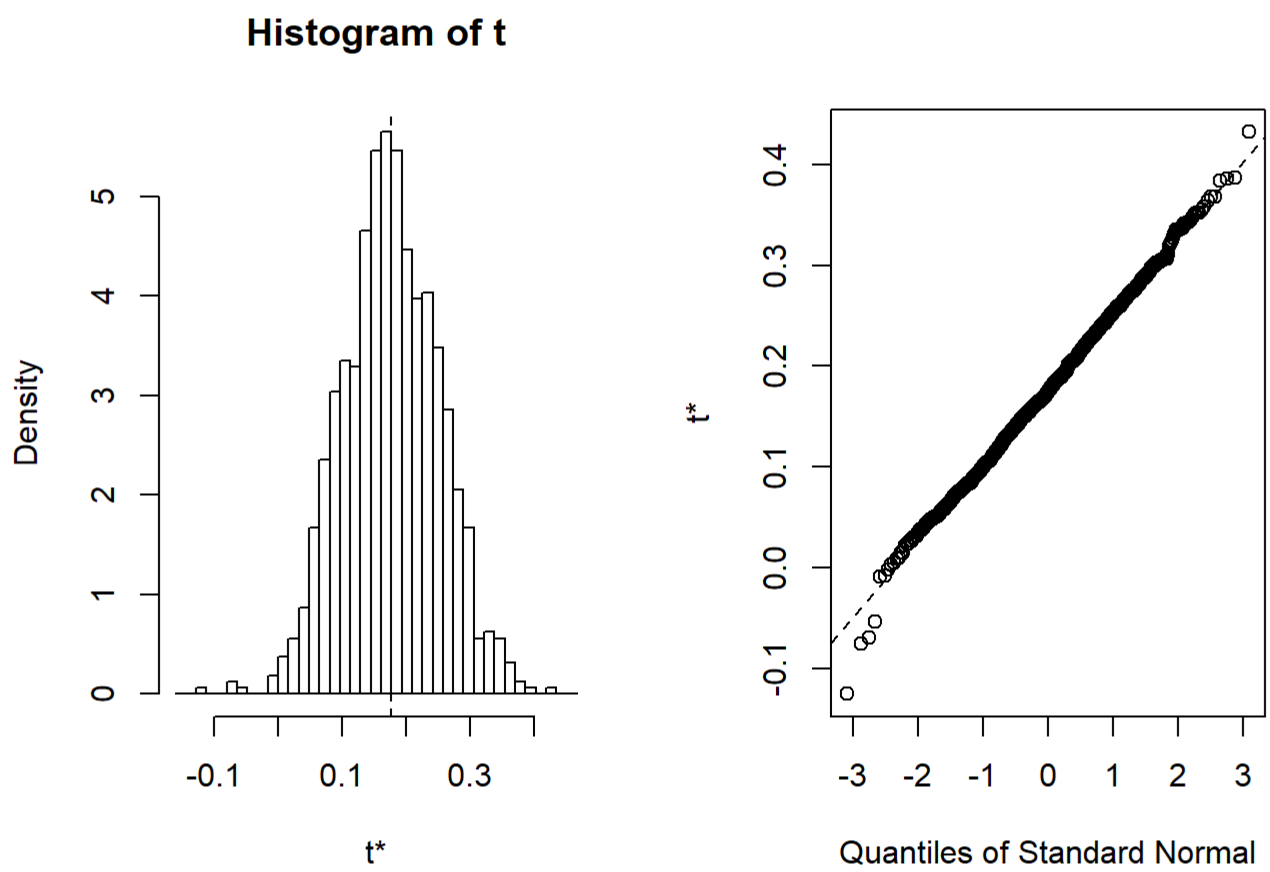

```
plot(amzn.jwnBootRho)
```



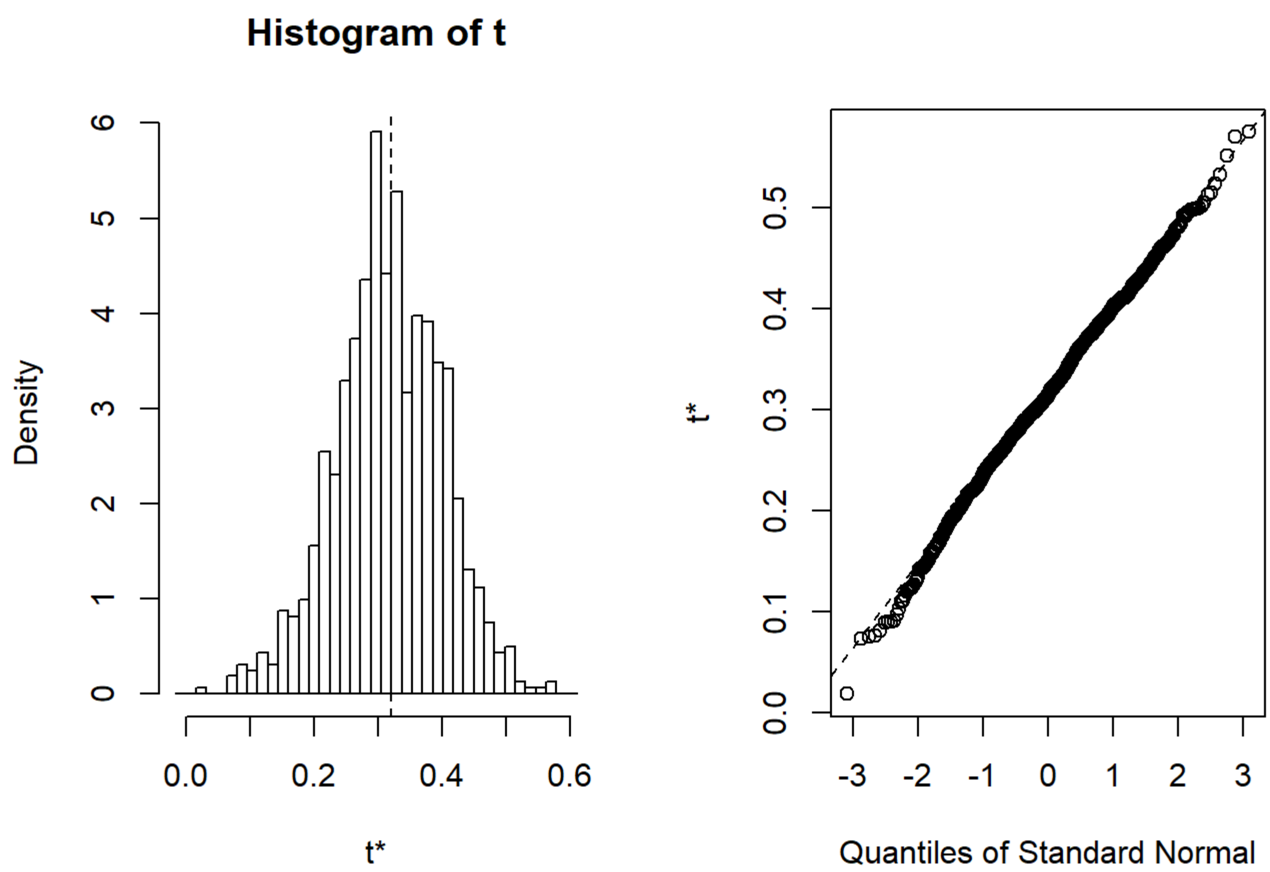

```
plot(amzn.sbuxBootRho)
```



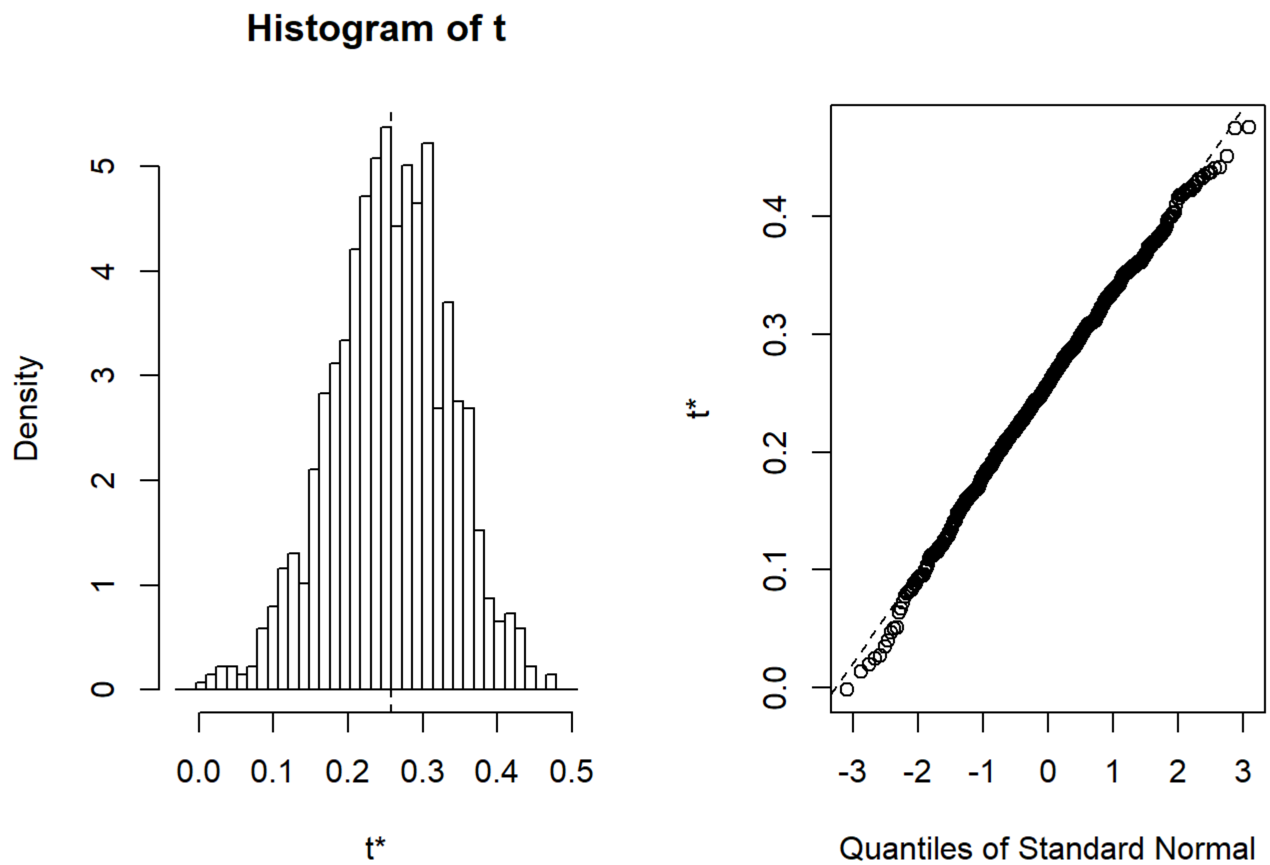

```
plot(ba.costBootRho)
```



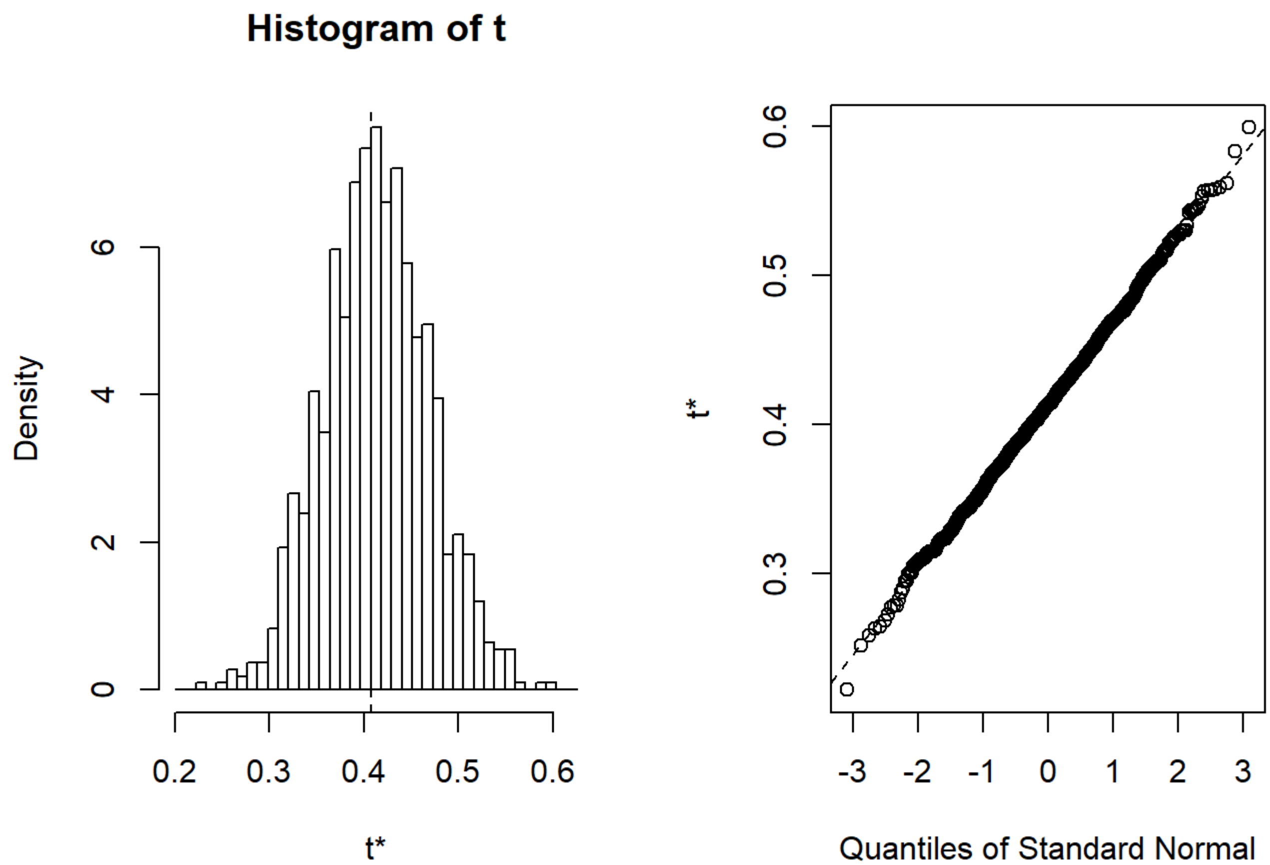

```
plot(ba.jwnBootRho)
```



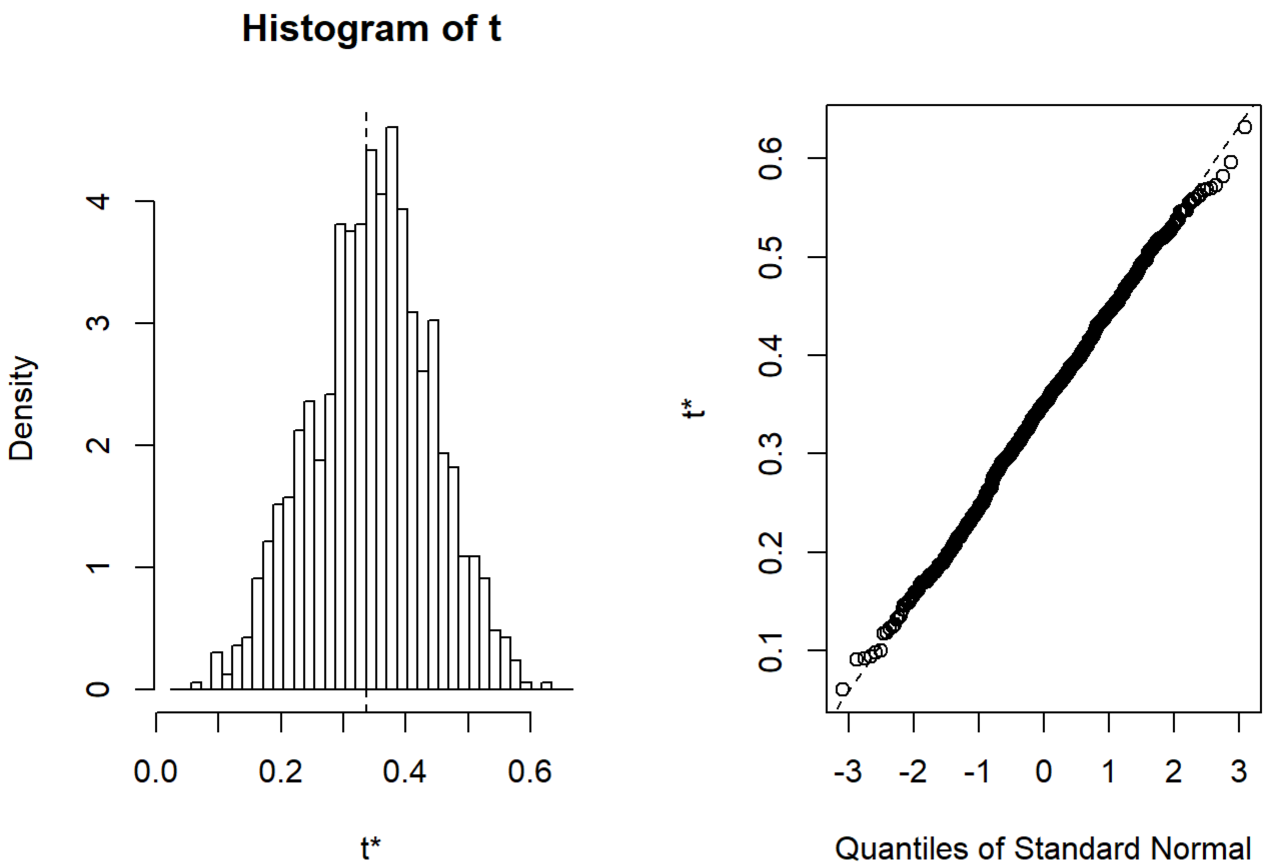

```
plot(ba.sbuxBootRho)
```



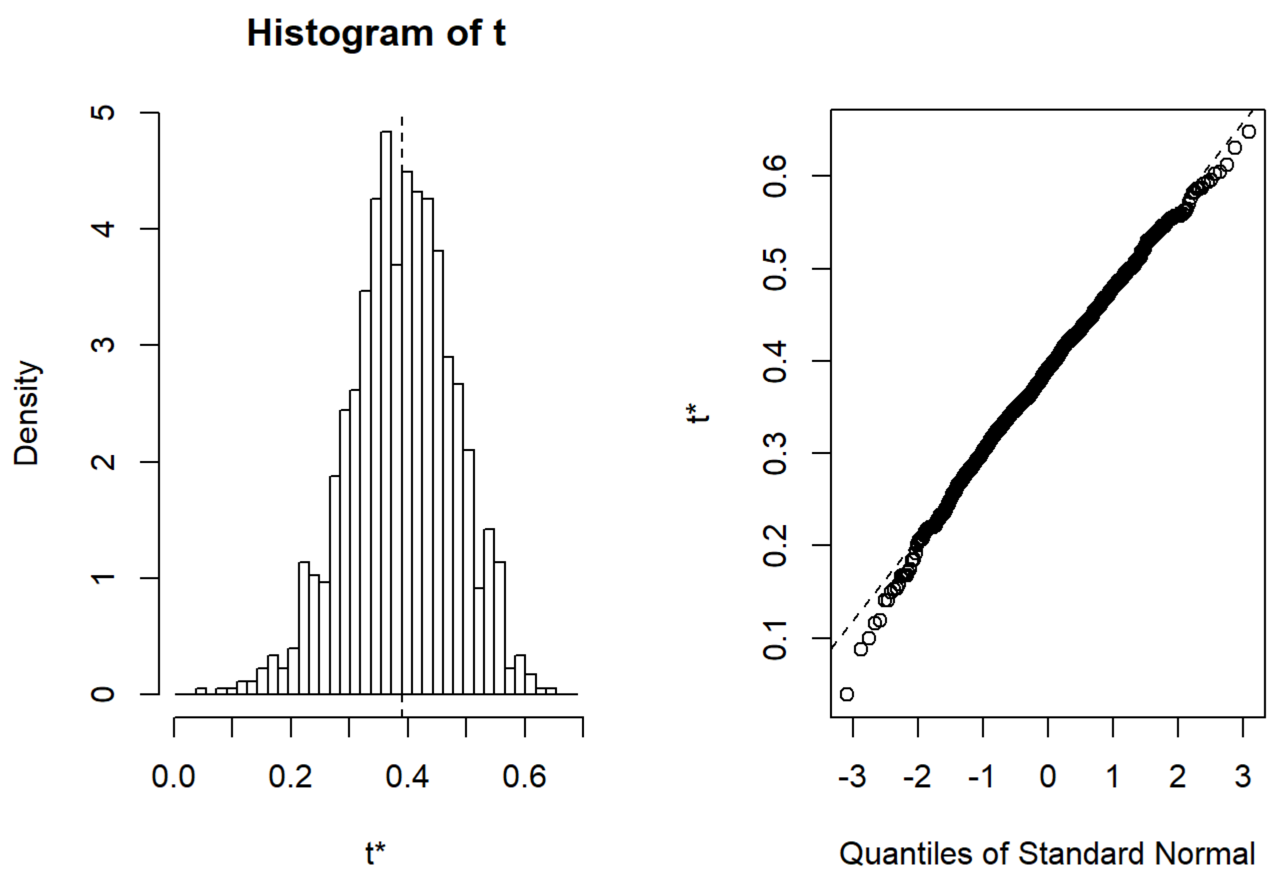

```
plot(cost.jwnBootRho)
```




```
plot(cost.sbuxBootRho)
```




```
plot(jwn.sbuxBootRho)
```



All of the bootstrap values that were calculated seem to be realtively normally distributed.

3. For each asset, compute estimates of the 5% value-at-risk based on an initial $\$100,000$ investment. Use the bootstrap to compute values as well as 95% confidence intervals. Briefly comment on the accuracy of the 5% VaR estimates.

```
ValueAtRisk.boot =function(x, idx, p=0.05, w=100000) {
  q = mean(x[idx]) + sd(x[idx]) * qnorm(p)
  VaR = (exp(q)-1) * w
  VaR
}

amzn.VaR.boot = boot(amznRet, ValueAtRisk.boot, 999)
ba.VaR.boot = boot(baRet, ValueAtRisk.boot, 999)
cost.VaR.boot = boot(costRet, ValueAtRisk.boot, 999)
jwn.VaR.boot = boot(jwnRet, ValueAtRisk.boot, 999)
sbux.VaR.boot = boot(sbuxRet, ValueAtRisk.boot, 999)
amzn.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = amznRet, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -22549     173        1872
```

```
ba.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = baRet, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -12817    38.3        1301
```

```
cost.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = costRet, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -10909    41.6        1677
```

```
jwn.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = jwnRet, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -16217    63.1        1546
```

```
sbux.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = sbuxRet, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -14680     129        1621
```

```
boot.ci(amzn.VaR.boot, 0.95, type = c("norm", "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = amzn.VaR.boot, conf = 0.95, type = c("norm",
##     "perc"))
##
## Intervals :
## Level      Normal             Percentile
## 95%   (-26391, -19053 )   (-26192, -18551 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(ba.VaR.boot, 0.95, type = c("norm", "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = ba.VaR.boot, conf = 0.95, type = c("norm",
##     "perc"))
##
## Intervals :
## Level      Normal             Percentile
## 95%   (-15405, -10305 )   (-15583, -10268 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(cost.VaR.boot, 0.95, type = c("norm", "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = cost.VaR.boot, conf = 0.95, type = c("norm",
##     "perc"))
##
## Intervals :
## Level      Normal             Percentile
## 95%   (-14238,  -7663 )   (-14679,  -8228 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(jwn.VaR.boot, 0.95, type = c("norm", "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = jwn.VaR.boot, conf = 0.95, type = c("norm",
##     "perc"))
##
## Intervals :
## Level      Normal             Percentile
## 95%   (-19310, -13250 )   (-19155, -13235 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(sbux.VaR.boot, 0.95, type = c("norm", "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = sbux.VaR.boot, conf = 0.95, type = c("norm",
##     "perc"))
##
## Intervals :
## Level      Normal             Percentile
## 95%   (-17987, -11632 )   (-17956, -11596 )
## Calculations and Intervals on Original Scale
```

Relative to the size of the investment, the ranges seem to be quite large telling us that these estimates are not very accurate.