Risk Analysis of IBM, GE and P&G

Deepti Gupta

July 11, 2019

Load the packages and libraries

```
#Install packages and include libraries

# install.packages(readxl)
# install.packages(FRAPO)
# install.packages(timeSeries)
# install.packages(QRM)
# install.packages(fGarch)
# install.packages(readr)
# install.packages(zoo)
# install.packages(fBasics)
# install.packages(evir)
# install.packages(ismev)
# install.packages(fExtremes)
# install.packages("GeneralizedHyperbolic")
# install.packages(ghyp)


getwd()
library(readxl)
library(FRAPO)
library(timeSeries)
library(QRM)
library(fGarch)
library(readr)
library(zoo)
library(fBasics)
library(evir)
library(ismev)
library(fExtremes)
library(GeneralizedHyperbolic)
library(ghyp)

setwd("C:/Study/515-RiskModelingAndAssessment/RStudio")
#setwd("C:/My/ANLY515")
```

Load Data - IBM, GE and PG, Frequency monthly, 1962-01-01 till current month

```
GE <- read_csv("./data/GE.csv",
    col_types = cols(`Adj Close` = col_number(),
        Close = col_number(), Date = col_date(format = "%Y-%m-%d"),
        High = col_number(), Low = col_number(),
        Open = col_number(), Volume = col_integer()))
```

```
## Warning: 19 parsing failures.
## row    col   expected     actual            file
## 561 Volume an integer 2230985900 './data/GE.csv'
## 562 Volume an integer 3842833000 './data/GE.csv'
## 563 Volume an integer 2603103000 './data/GE.csv'
## 564 Volume an integer 2195174000 './data/GE.csv'
## 565 Volume an integer 2335377200 './data/GE.csv'
## ... ...... .......... .......... ..............
## See problems(...) for more details.
```

```
head(GE)
```

```
## # A tibble: 6 x 7
##   Date        Open  High   Low Close `Adj Close`   Volume
##   <date>     <dbl> <dbl> <dbl> <dbl>       <dbl>    <int>
## 1 1962-01-01 0.751 0.764 0.691 0.751       0.126 39894800
## 2 1962-02-01 0.751 0.781 0.736 0.747       0.126 28463300
## 3 1962-03-01 0.751 0.786 0.751 0.770       0.129 28882600
## 4 1962-04-01 0.769 0.769 0.676 0.686       0.116 34393600
## 5 1962-05-01 0.686 0.724 0.601 0.659       0.111 60641700
## 6 1962-06-01 0.659 0.665 0.543 0.596       0.101 50248400
```

```
IBM <- read_csv("./data/IBM.csv",
    col_types = cols(`Adj Close` = col_number(),
        Close = col_number(), Date = col_date(format = "%Y-%m-%d"),
        High = col_number(), Low = col_number(),
        Open = col_number(), Volume = col_integer()))
head(IBM)
```

```
## # A tibble: 6 x 7
##   Date        Open  High   Low Close `Adj Close`   Volume
##   <date>     <dbl> <dbl> <dbl> <dbl>       <dbl>    <int>
## 1 1962-01-01  7.71  7.71  7.00  7.23        1.89  8760000
## 2 1962-02-01  7.3   7.48  7.09  7.16        1.87  5737600
## 3 1962-03-01  7.19  7.41  7.07  7.10        1.86  5344000
## 4 1962-04-01  7.1   7.1   6     6.05        1.58 12851200
## 5 1962-05-01  6.05  6.53  4.73  5.23        1.37 49307200
## 6 1962-06-01  5.21  5.21  4     4.52        1.18 68451200
```

```
PG <- read_csv("./data/PG.csv",
    col_types = cols(`Adj Close` = col_number(),
        Close = col_number(), Date = col_date(format = "%Y-%m-%d"),
        High = col_number(), Low = col_number(),
        Open = col_number(), Volume = col_integer()))
head(PG)
```

```
## # A tibble: 6 x 7
##   Date        Open  High   Low Close `Adj Close`   Volume
##   <date>     <dbl> <dbl> <dbl> <dbl>       <dbl>    <int>
## 1 1962-01-01  1.42  1.44 1.27  1.29      0.0124   9670400
## 2 1962-02-01  1.29  1.30 1.24  1.24      0.0119   7155200
## 3 1962-03-01  1.25  1.35 1.24  1.34      0.0129   7859200
## 4 1962-04-01  1.34  1.35 1.25  1.26      0.0121   6937600
## 5 1962-05-01  1.26  1.30 1.00  1.13      0.0109  11936000
## 6 1962-06-01  1.13  1.13 0.881 0.977     0.00942 12428800
```

Pre-process the data

```
IBMPrice <- IBM$Close
GEPrice  <- GE$Close
PGPrice  <- PG$Close

date <- as.character(IBM$Date)
dateSub  <- date[date > "1962-01-01"]

attr(IBMPrice, 'time') <- date
attr(GEPrice , 'time') <- date
attr(PGPrice , 'time') <- date

IBMRet <- na.omit(returnSeries(IBMPrice))
GERet  <- na.omit(returnSeries(GEPrice))
PGRet  <- na.omit(returnSeries(PGPrice) )

attr(IBMRet, 'time') <- dateSub
attr(GERet , 'time') <- dateSub
attr(PGRet , 'time') <- dateSub


##Calculate losses
IBMloss <- as.data.frame(na.omit(-1.0*diff(log(IBM$Close))*100.0))
colnames(IBMloss) <- c("IBM")
head(IBMloss)
```

```
##           IBM
## 1   0.9267815
## 2   0.7945870
## 3  15.9955081
## 4  14.5560669
## 5  14.5799027
## 6 -13.1687247
```

```
GEloss <- as.data.frame(na.omit(-1.0*diff(log(GE$Close))*100.0))
colnames(GEloss) <- c("GE")
head(GEloss)
```

```
##            GE
## 1   0.5012528
## 2  -2.9705076
## 3  11.5346663
## 4   4.0973952
## 5   9.9883039
## 6 -10.9343819
```

```
PGloss <- as.data.frame(na.omit(-1.0*diff(log(PG$Close))*100.0))
colnames(PGloss) <- c("PG")
head(PGloss)
```

```
##           PG
## 1   4.006678
## 2  -7.567885
## 3   6.007811
```

```
## 4  10.777116
## 5  14.841993
## 6 -10.616014
```

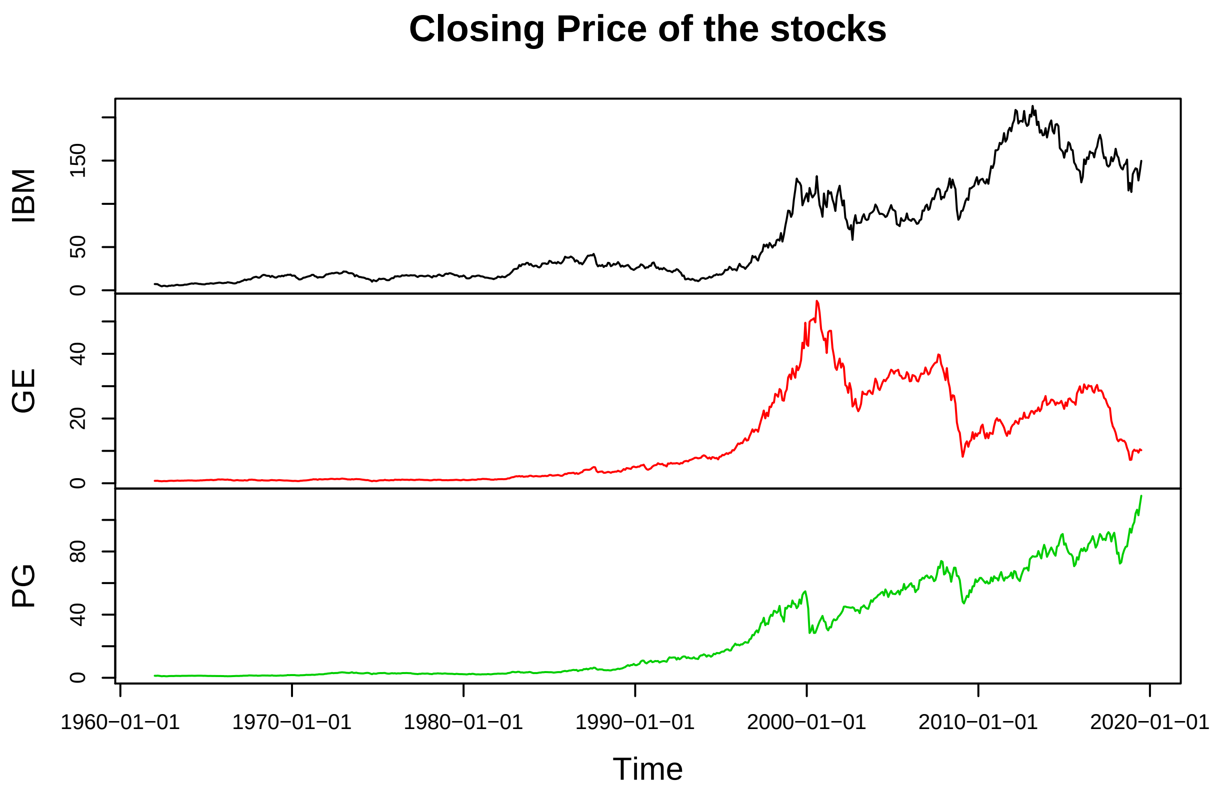
```r
IBMLossTs <- timeSeries(IBMloss$IBM, charvec = dateSub)
GELossTs  <- timeSeries(GEloss$GE,   charvec = dateSub)
PGLossTs  <- timeSeries(PGloss$PG,   charvec = dateSub)


dataset <- cbind(IBM$Close, GE$Close, PG$Close )
colnames(dataset) <- c("IBM", "GE", "PG")
head(dataset)
```

```
##           IBM       GE       PG
## [1,] 7.226666 0.751202 1.292969
## [2,] 7.160000 0.747446 1.242188
## [3,] 7.103333 0.769982 1.339844
## [4,] 6.053333 0.686098 1.261719
## [5,] 5.233333 0.658554 1.132813
## [6,] 4.523334 0.595954 0.976563
```

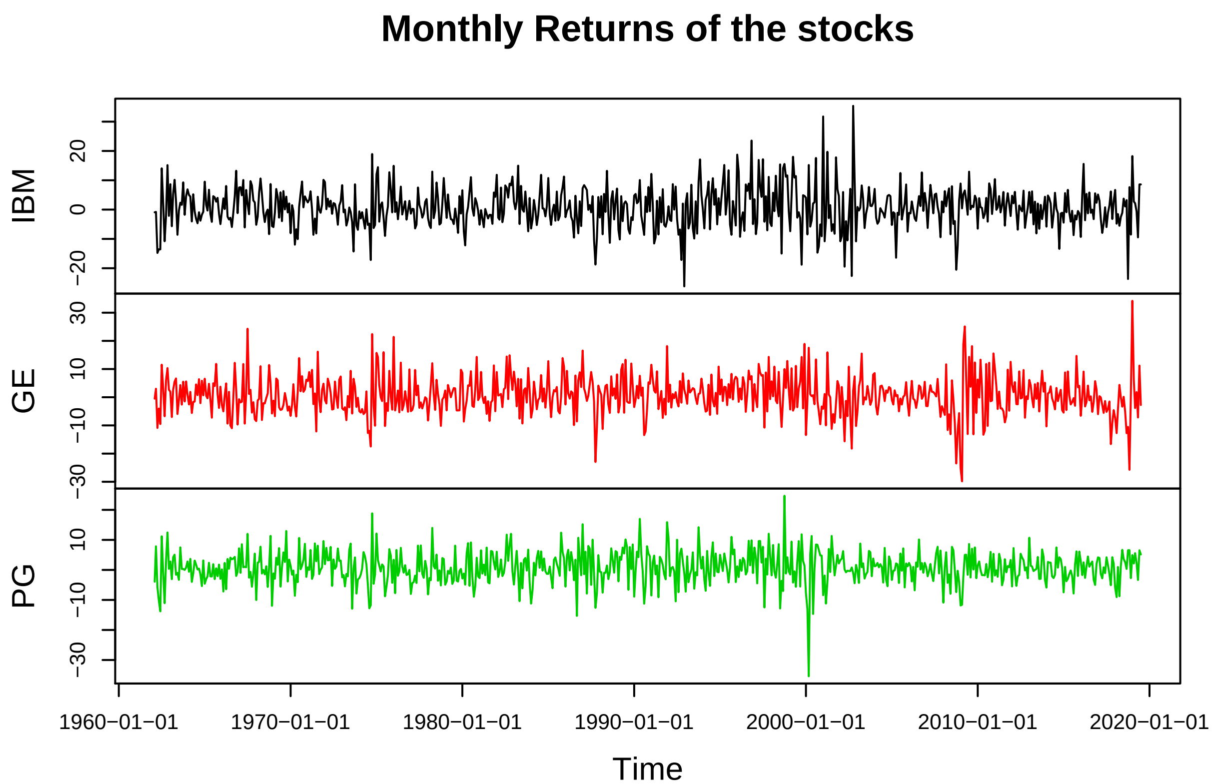
```r
dataTS <- timeSeries(dataset[, c("IBM","GE", "PG")], charvec = date)
head(dataTS)
```

```
## GMT
##                 IBM       GE       PG
## 1962-01-01 7.226666 0.751202 1.292969
## 1962-02-01 7.160000 0.747446 1.242188
## 1962-03-01 7.103333 0.769982 1.339844
## 1962-04-01 6.053333 0.686098 1.261719
## 1962-05-01 5.233333 0.658554 1.132813
## 1962-06-01 4.523334 0.595954 0.976563
```

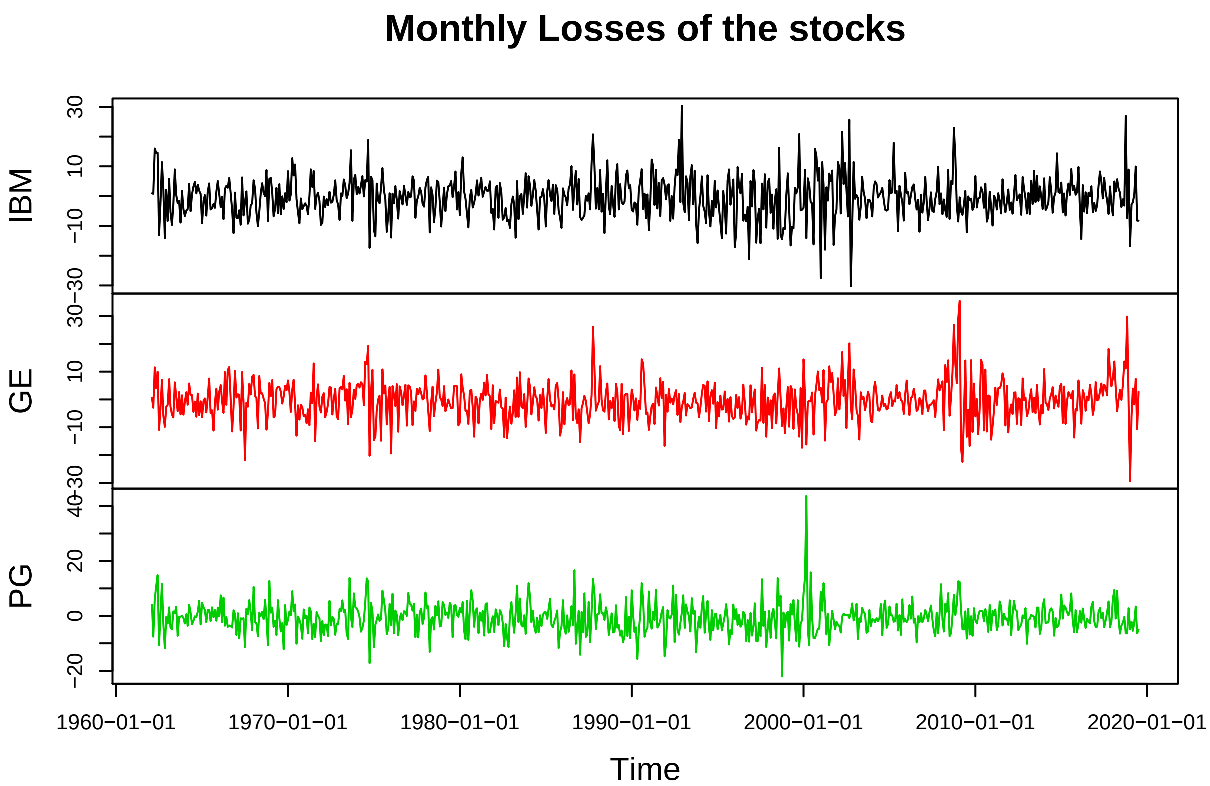
```r
plot(dataTS, main = "Closing Price of the stocks")
```




```
dataReturns <- na.omit(returnseries(dataTS, method = "discrete",trim = FALSE))
plot(dataReturns, main = "Monthly Returns of the stocks")
```




```
dataloss <- as.data.frame(na.omit(-1.0*diff(log(dataset))*100.0))
datalossts <- timeSeries(dataloss, charvec = dateSub)
plot(datalossts, main = "Monthly Losses of the stocks")
```



Covariance And Global Minimum Variance Portfolio

```r
# covariance matrix using cov() function and "pairwise.complete.obs" specification
dataCOV <- cov(dataReturns, use="pairwise.complete.obs")
dataCOV
```

```
##            IBM       GE        PG
## IBM 48.155056 21.05536  8.395876
## GE  21.055361 49.35566 15.337801
## PG   8.395876 15.33780 29.891785
```

```r
#Find weights of the "global minimum variance portfolio".
PGMV<-PGMV(dataCOV)
```

```
## Iteration: 0
## pobj: 0.00015909
## dobj: -1.14153
## pinf: 0
## dinf: 2.72298
## dgap: 1.14169
##
## Iteration: 1
## pobj: 7.90539e-007
## dobj: -0.0587284
## pinf: 1.11022e-016
## dinf: 0.139415
## dgap: 0.0587292
##
```

```
## Iteration: 2
## pobj: 2.07766e-009
## dobj: -0.00294269
## pinf: 2.22045e-016
## dinf: 0.00698218
## dgap: 0.00294269
##
## Iteration: 3
## pobj: 5.20792e-012
## dobj: -0.000147151
## pinf: 4.44089e-016
## dinf: 0.000349138
## dgap: 0.000147151
##
## Iteration: 4
## pobj: 1.6495e-014
## dobj: -7.35758e-006
## pinf: 7.77156e-016
## dinf: 1.7457e-005
## dgap: 7.35758e-006
##
## Iteration: 5
## pobj: 2.55774e-016
## dobj: -3.67879e-007
## pinf: 1.55431e-015
## dinf: 8.7285e-007
## dgap: 3.67879e-007
##
## Iteration: 6
## pobj: 4.60066e-015
## dobj: -1.83939e-008
## pinf: 3.10862e-015
## dinf: 4.36425e-008
## dgap: 1.8394e-008
##
## Optimal solution found.
```

```
PGMV
```

```
##
##
## Optimal weights for porfolio of type:
## Global Minimum Variance
##
##     IBM      GE      PG
## 28.9986 13.7167 57.2847
```

```
w<-Weights(PGMV)/100
wIBM <- as.numeric(w[1])
wGE  <- as.numeric(w[2])
wPG  <- as.numeric(w[3])

dataCOV
```

```
##           IBM       GE        PG
## IBM 48.155056 21.05536  8.395876
```

```
## GE  21.055361 49.35566 15.337801
## PG   8.395876 15.33780 29.891785
```

```r
wIBM
```

```
## [1] 0.289986
```

```r
wGE
```

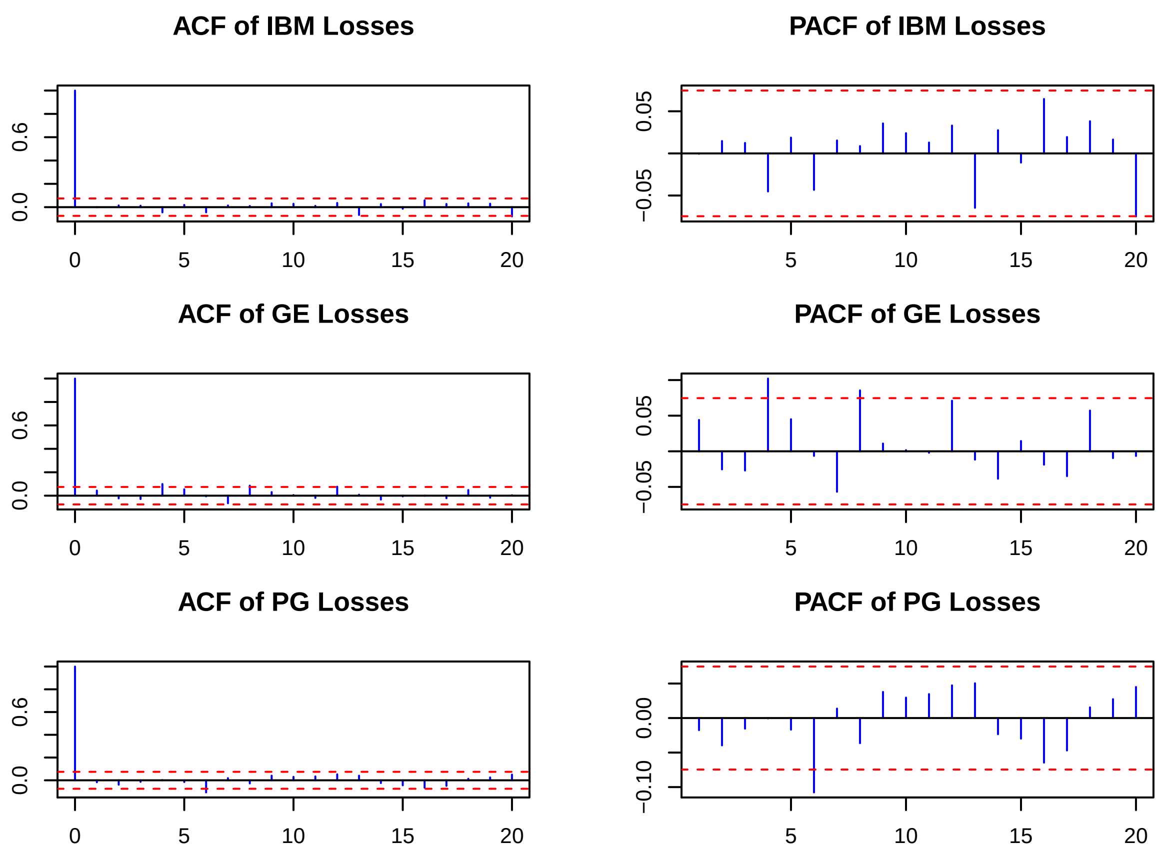
```
## [1] 0.1371671
```

```r
wPG
```

```
## [1] 0.5728469
```

Check ACF and Test Auto-correlation

```r
par(mfrow=c(3,2), mar = c(3,3,3,3) )
acf(IBMloss$IBM, main="ACF of IBM Losses" , lag.max=20, ylab="", xlab= "", col="blue", ci.col="red")
pacf(IBMloss$IBM, main="PACF of IBM Losses", lag.max=20,ylab="", xlab = "", col = "blue", ci.col="red")

acf(GEloss$GE, main="ACF of GE Losses" , lag.max=20, ylab="", xlab= "", col="blue", ci.col="red")
pacf(GEloss$GE, main="PACF of GE Losses", lag.max=20,ylab="", xlab = "", col = "blue", ci.col="red")

acf(PGloss$PG, main="ACF of PG Losses" , lag.max=20, ylab="", xlab= "", col="blue", ci.col="red")
pacf(PGloss$PG, main="PACF of PG Losses" , lag.max=20,ylab="", xlab = "", col = "blue", ci.col="red")
```




```r
Box.test(IBMloss$IBM, lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
```

```
##
## data:  IBMloss$IBM
## X-squared = 4.8673, df = 10, p-value = 0.8999
```

```r
# There is one autocorrelation lying outside the 95% limits, and the Ljung-Box
# statistic has a p-value of 0.8 (for  h = 10  ). This suggests that the monthly
# change in the IBM stock price is essentially a random amount which is uncorrelated
# with that of previous month

Box.test(GEloss$GE, lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  GEloss$GE
## X-squared = 20.165, df = 10, p-value = 0.02773
```

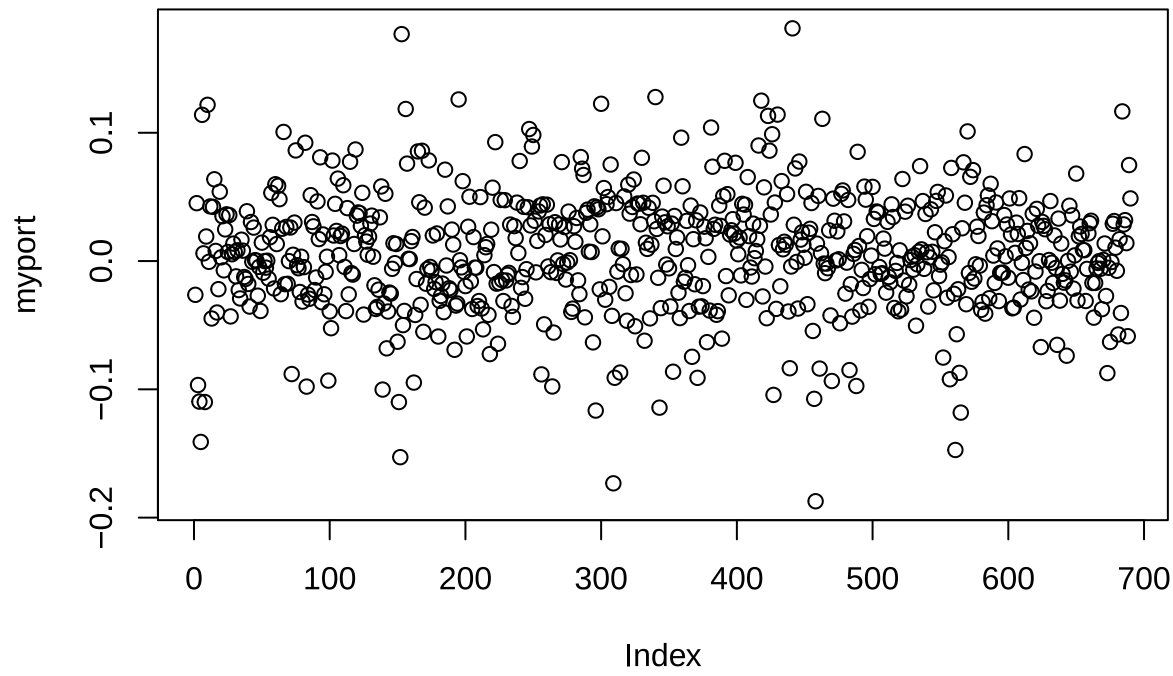
```r
# There is one autocorrelation lying outside the 95% limits, and the Ljung-Box
# statistic has a p-value of 0.02 (for  h = 10), suggesting that the monthly
# change in the GE stock price is correlated with that of previous month


Box.test(PGloss$PG, lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  PGloss$PG
## X-squared = 12.127, df = 10, p-value = 0.2767
```

```r
# There is one autocorrelation lying outside the 95% limits, and the Ljung-Box
# statistic has a p-value of 0.2 (for  h = 10 ). This suggests that the monthly
# change in the PG stock price is essentially a random amount which is uncorrelated
# with that of previous month
```

METHOD 1: Fitting generalized hyperbolic distribution model and Calculating Risks

```r
myport <- (wIBM * IBMRet) + (wGE * GERet) + (wPG * PGRet)
plot(myport)
```




```
myportts <- timeSeries(myport, charvec = dateSub)
str(myportts)
```

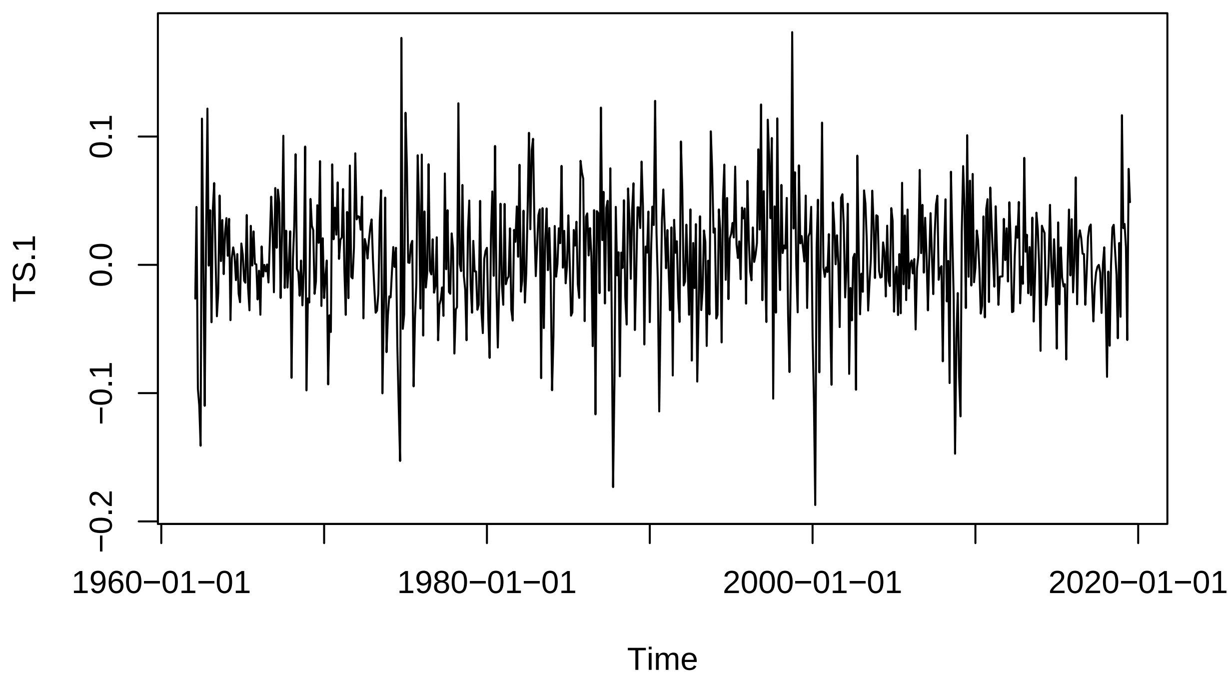
```
## Time Series:
##  Name:               object
## Data Matrix:
##  Dimension:          690 1
##  Column Names:       TS.1
##  Row Names:          1962-02-01  ...  2019-07-01
## Positions:
##  Start:              1962-02-01
##  End:                2019-07-01
## With:
##  Format:             %Y-%m-%d
##  FinCenter:          GMT
##  Units:              TS.1
##  Title:              Time Series Object
##  Documentation:      Thu Aug 08 21:34:04 2019
```

```
head(myportts)
```

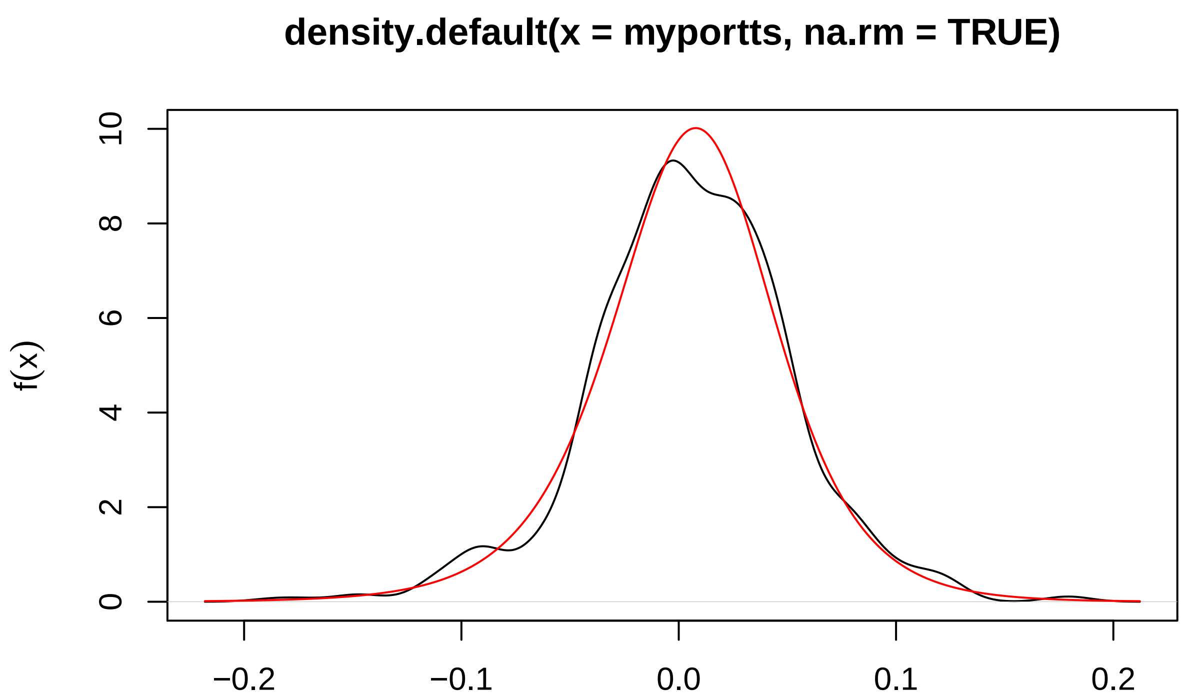
```
## GMT
##                  TS.1
## 1962-02-01 -0.02632722
## 1962-03-01  0.04512276
## 1962-04-01 -0.09662206
## 1962-05-01 -0.10956720
```

```
## 1962-06-01 -0.14100224
## 1962-07-01  0.11399934
```

```r
plot(myportts) # high volatility observed
```



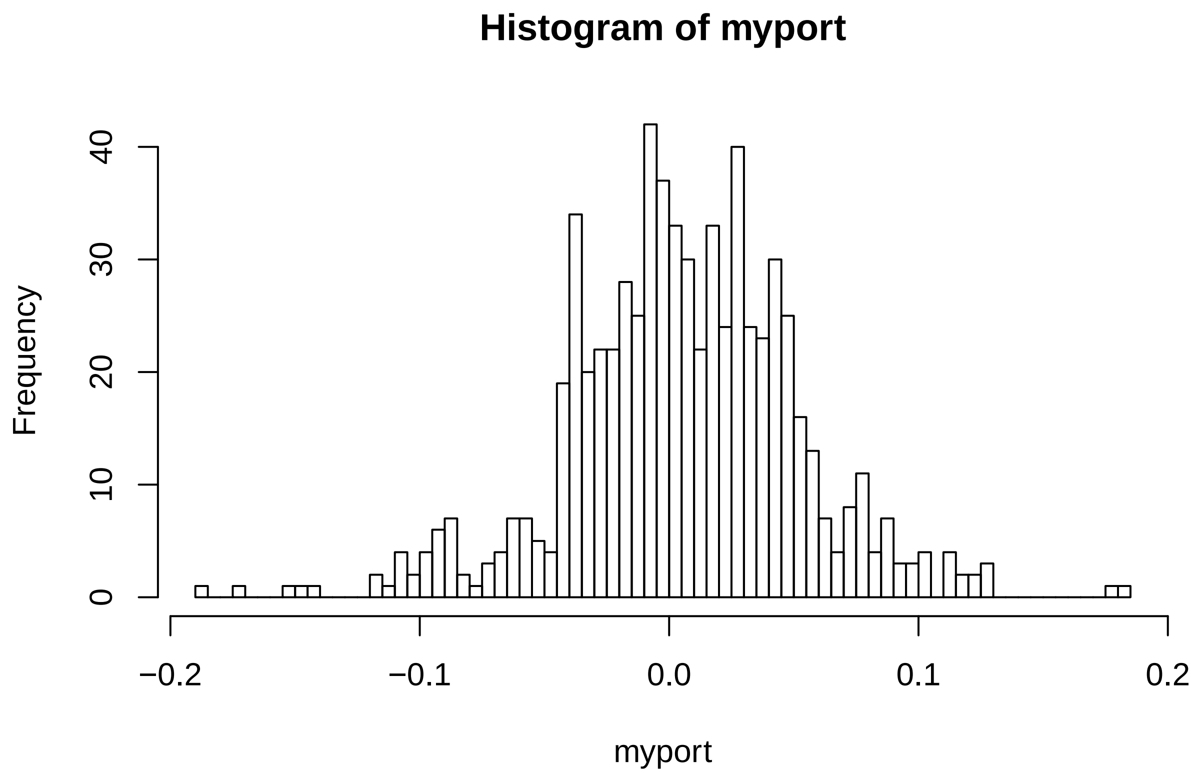

```r
# Use stepAIC.ghyp() function to see which distribution has the closest fit to the actual distribution


AIC <- stepAIC.ghyp(myportts)
AIC$fit.table

# Goal is aic should be minimum which is true for ghyp (generalized hyperbolic distribution)  model for

# Fit "myportts" data to the chosen model and save the estimated coeficients as xxxfit.
# where xxx represents the choice of the model ("ghyp", "hyp", "NIG", "VG", "t", "gauss")

ghypfit<- fit.ghypuv(myportts, symmetric = FALSE, control = list(maxit = 1000), na.rm = TRUE)


#
ef <- density(myportts, na.rm = TRUE)
ghypdens <- dghyp(ef$x , ghypfit)

plot(ef, xlab = "", ylab = expression(f(x)), ylim = c(0, 10))
lines(ef$x , ghypdens, col = "red")
```




```r
summary(ghypfit)
```

```
## Asymmetric Generalized Hyperbolic Distribution:
##
## Parameters:
##       lambda    alpha.bar           mu        sigma        gamma
## -1.281360822  1.824255192  0.010769568  0.046443330 -0.005246115
##
## Call:
## fit.ghypuv(data = myportts, symmetric = FALSE, na.rm = TRUE,     control = list(maxit = 1000))
##
## Optimization information:
## log-Likelihood:                 1152.175
## AIC:                            -2294.349
## Fitted parameters:              lambda, alpha.bar, mu, sigma, gamma;  (Number: 5)
## Number of iterations:           590
## Converged:                      TRUE
```

```r
hist(myport, breaks = 100) #left tail for VaR and right tail for ES
```




```r
p <- seq(0.01,0.05,0.01)

portvar <- abs(qghyp(p, ghypfit)) * 100
portvar          # VaR Values in the vector form of quantile
```

```
## [1] 11.918186  9.834482  8.630620  7.780376  7.121447
```

```r
# 99%    98%    97%    96%    95%
# 11.9   9.8    8.6    7.7    7.1

portes <- abs(ESghyp(p, ghypfit)) * 100
portes           # ES values in the vector form of quantile
```

```
## [1] 15.04814 12.90098 11.66441 10.79456 10.12338
```

```r
# 99%    98%    97%    96%    95%
# 15.04  12.9   11.66  10.7   10.12
```

METHOD 2 - MODEL THE LOSSES FOR GARCH AND ARIMA

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.5.3

##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:nlme':
##
##     getResponse
```

```
auto.arima(IBMloss$IBM)
```

```
## Series: IBMloss$IBM
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##          mean
##       -0.4393
## s.e.   0.2629
##
## sigma^2 estimated as 47.76:  log likelihood=-2312.42
## AIC=4628.83   AICc=4628.85   BIC=4637.91
```

```
auto.arima(GEloss$GE)
```

```
## Series: GEloss$GE
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##           ar1      ar2     ma1     ma2     mean
##       -0.2559  -0.9143  0.3152  0.9286  -0.3758
## s.e.   0.0505   0.0419  0.0425  0.0444   0.2733
##
## sigma^2 estimated as 48.56:  log likelihood=-2316.25
## AIC=4644.5   AICc=4644.62   BIC=4671.72
```

```
auto.arima(PGloss$PG)
```

```
## Series: PGloss$PG
## ARIMA(2,0,3) with non-zero mean
##
## Coefficients:
```

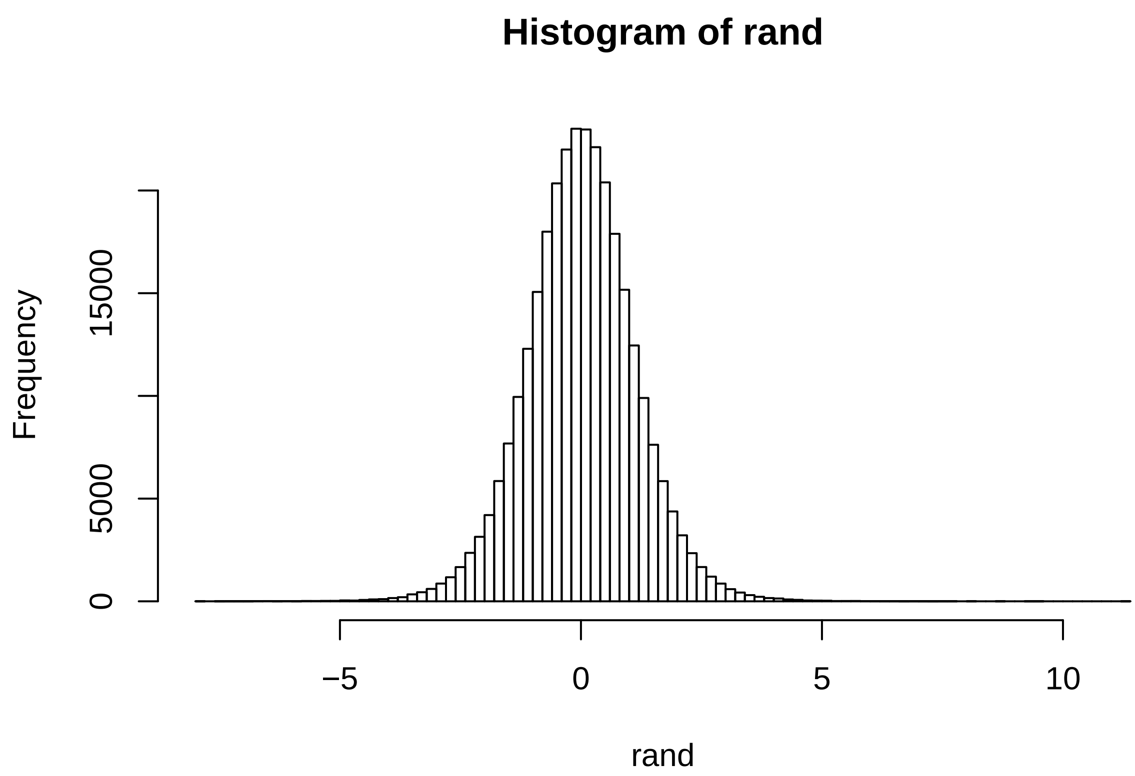
```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##           ar1      ar2     ma1     ma2      ma3     mean
##       -0.0785  -0.8743  0.0602  0.8438  -0.0212  -0.6462
## s.e.   0.0953      NaN  0.1022     NaN   0.0395   0.2022
##
## sigma^2 estimated as 30.61:  log likelihood=-2156.46
## AIC=4326.93   AICc=4327.09   BIC=4358.68
```

```
## Step 1 : Estimate ARMA+GARCH model and create 1 step ahead forecast of the stdev
gafit <- lapply(dataloss, garchFit , formula =  ~arma(0,0) + garch(1, 1) ,cond.dist="std",trace=FALSE)

gaprog <- unlist(lapply(gafit , function(x) predict(x,n.ahead = 1)[3]))


##Step 2 : Find the best fitting distribution and estimate distribution coefficients for the return seri
df <- unlist(lapply(gafit, function(x) x@fit$coef[5]))
head(df)
```

```
## IBM.shape  GE.shape  PG.shape
##  8.137524 10.000000  9.049799
```

```
## Step 3: Simulate 100,000 random variables(returns) that follow the identified distribution with esti

rand <- sapply(1:3, function(x) rt(100000, df = df[x]) )
hist(rand, breaks = 100)
```




```
ht.mat <- matrix(gaprog, nrow = 100000, ncol = ncol(dataloss), byrow = TRUE)
head(ht.mat)
```

```
##          [,1]     [,2]     [,3]
## [1,] 8.10235 10.08633 4.872721
## [2,] 8.10235 10.08633 4.872721
## [3,] 8.10235 10.08633 4.872721
## [4,] 8.10235 10.08633 4.872721
## [5,] 8.10235 10.08633 4.872721
## [6,] 8.10235 10.08633 4.872721
```

```
## Step 4: Multiply each simulated return by forecasted stdev
weights <- c(wIBM, wGE , wPG )    # from GMVP portfolio

pfall.garch <- (rand * ht.mat) %*% weights
head(pfall.garch)
```

```
##             [,1]
## [1,]  2.02576189
## [2,]  6.24302011
## [3,] -0.03829155
## [4,]  2.65353306
```

```
## [5,]  0.93189358
## [6,] -0.68696644
```

```r
## Step 5: Sort the product obtained in 4 from smallest to largest
#a. VAR is the 5,000th largest loss
#b. ES is the median of 5,000 largest losses
pfall.garch.es95 <- median(tail(sort(pfall.garch), 5000))
pfall.garch.es95    #8.787554
```

```
## [1] 8.780425
```

```r
pfall.garch.var95 <- min(tail(sort(pfall.garch), 5000))
pfall.garch.var95     #7.234231
```

```
## [1] 7.274805
```

METHOD 3 - USING GARCH-COPULA APPROACH TO DETERMINE PORTFOLIO RISK

```r
head(dataset)
```

```
##           IBM       GE       PG
## [1,] 7.226666 0.751202 1.292969
## [2,] 7.160000 0.747446 1.242188
## [3,] 7.103333 0.769982 1.339844
## [4,] 6.053333 0.686098 1.261719
## [5,] 5.233333 0.658554 1.132813
## [6,] 4.523334 0.595954 0.976563
```

```r
head(dataloss)
```

```
##           IBM          GE         PG
## 1   0.9267815   0.5012528   4.006678
## 2   0.7945870  -2.9705076  -7.567885
## 3  15.9955081  11.5346663   6.007811
## 4  14.5560669   4.0973952  10.777116
## 5  14.5799027   9.9883039  14.841993
## 6 -13.1687247 -10.9343819 -10.616014
```

```r
### Step 1:  Estimate GARCH model
dfit<-lapply(dataloss,garchFit,formula=~arma(0,0)+garch(1,1),cond.dist="std",trace=FALSE)
dfit
```

```
## $IBM
##
## Title:
##  GARCH Modelling
##
## Call:
##  FUN(formula = ..1, data = X[[i]], cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 1)
## <environment: 0x0000000021121920>
##  [data = X[[i]]]
##
```

```
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        mu      omega     alpha1      beta1      shape
## -0.528561   3.097037   0.090824   0.843580   8.137524
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.52856     0.23201   -2.278 0.022714 *
## omega    3.09704     1.50643    2.056 0.039794 *
## alpha1   0.09082     0.03062    2.966 0.003014 **
## beta1    0.84358     0.05286   15.958  < 2e-16 ***
## shape    8.13752     2.14862    3.787 0.000152 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2274.142    normalized:  -3.295858
##
## Description:
##  Thu Aug 08 21:34:07 2019 by user: rgoyal
##
##
## $GE
##
## Title:
##  GARCH Modelling
##
## Call:
##  FUN(formula = ..1, data = X[[i]], cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 1)
## <environment: 0x000000001d5e4128>
##  [data = X[[i]]]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##       mu     omega    alpha1     beta1     shape
## -0.64770   2.77005   0.13010   0.81772  10.00000
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.64770     0.22863   -2.833 0.004611 **
## omega    2.77005     1.17891    2.350 0.018790 *
```

```
## alpha1   0.13010     0.03256    3.996 6.45e-05 ***
## beta1    0.81772     0.04112   19.886  < 2e-16 ***
## shape   10.00000     2.94747    3.393 0.000692 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2270.571    normalized:  -3.290683
##
## Description:
##  Thu Aug 08 21:34:07 2019 by user: rgoyal
##
##
## $PG
##
## Title:
##  GARCH Modelling
##
## Call:
##  FUN(formula = ..1, data = X[[i]], cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 1)
## <environment: 0x00000000249737e8>
##  [data = X[[i]]]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##       mu     omega    alpha1     beta1     shape
## -0.72225   3.01425   0.10083   0.79462   9.04980
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      -0.72225     0.18570   -3.889 0.000101 ***
## omega    3.01425     1.26940    2.375 0.017571 *
## alpha1   0.10083     0.03554    2.837 0.004559 **
## beta1    0.79462     0.06278   12.657  < 2e-16 ***
## shape    9.04980     2.72728    3.318 0.000906 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2116.157    normalized:  -3.066895
##
## Description:
##  Thu Aug 08 21:34:07 2019 by user: rgoyal
```

```r
dprog<-unlist(lapply(dfit,function(x) predict(x,n.ahead = 1)[3]))
```

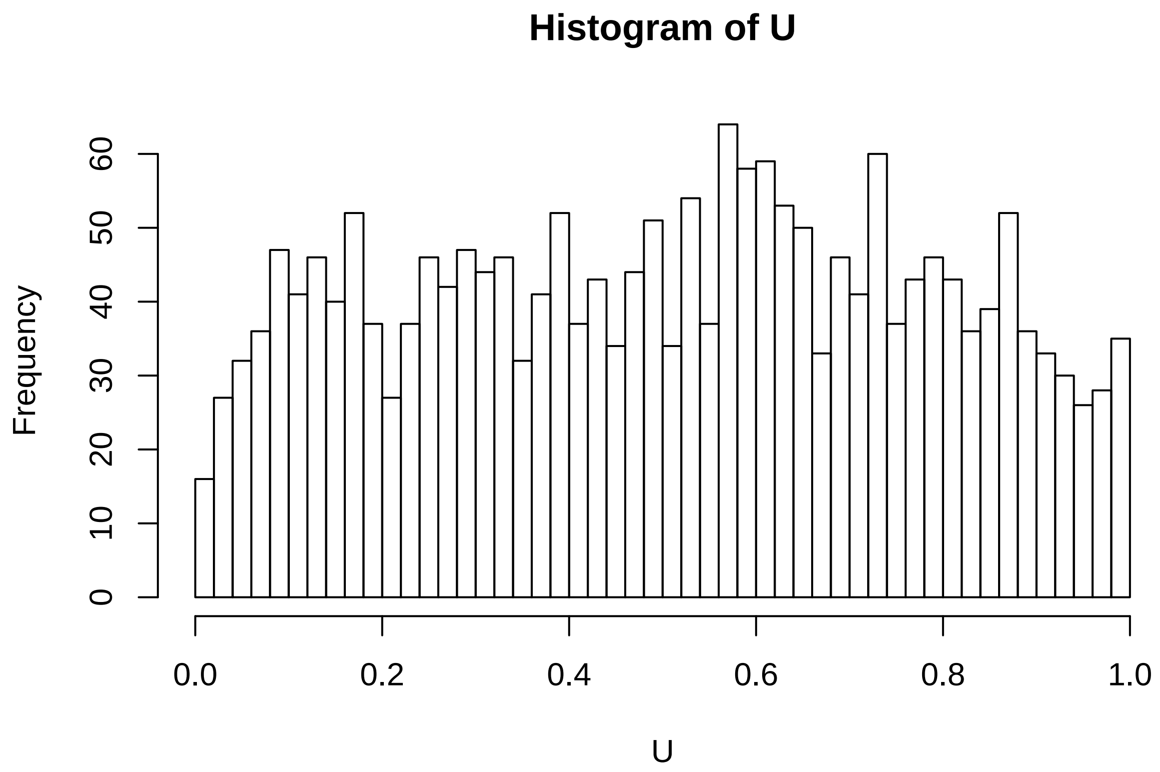
```r
# Estimate degrees-of-freedom parameters
dshape<-unlist(lapply(dfit, function(x) x@fit$coef[5]))

### Step 2: Estimates conditional standardized residuals are extracted.(h.t - conditional variance)
dresid<-as.matrix(data.frame(lapply(dfit,function(x) x@residuals / sqrt(x@h.t))))
head(dresid)
```

```
##             IBM         GE         PG
## [1,]  0.2107890  0.1632255  0.8591323
## [2,]  0.2006108 -0.3523522 -1.2637342
## [3,]  2.6142078  1.9505238  1.2076588
## [4,]  1.9219594  0.6458657  2.0246475
## [5,]  1.7361662  1.5066150  2.4021745
## [6,] -1.3496566 -1.3479162 -1.2688773
```

```r
### Step 3 : pseudo-uniform variables that generates probabilites for each risk from the standardized re
U <- sapply(1:3, function(y) pt(dresid[, y], df = dshape[y]))
head(U)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.5808826 0.5632038 0.7937772
## [2,] 0.5770369 0.3659435 0.1189533
## [3,] 0.9847556 0.9601605 0.8711042
## [4,] 0.9548882 0.7335468 0.9633025
## [5,] 0.9399455 0.9185856 0.9801935
## [6,] 0.1067344 0.1037128 0.1180739
```

```r
hist(U, breaks = 50)
```




```
### Step 4 : Estimate the copula model - Student's t copula is estimated based on Kendall's rank correla

cop <- fit.tcopula(Udata = U, method = "Kendall")
```

```
## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in dmt(Qdata, df, rep(0, d), Sigma, log = TRUE): value out of range
## in 'lgamma'

## Warning in dmt(Qdata, df, rep(0, d), Sigma, log = TRUE): value out of range
## in 'lgamma'

## Warning in log(pi * df): NaNs produced

## Warning in nlminb(startdf, negloglik2, data = Udata, P = P, ...): NA/NaN
## function evaluation

## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in FUN(newX[, i], ...): NaNs produced

## Warning in dmt(Qdata, df, rep(0, d), Sigma, log = TRUE): value out of range
## in 'lgamma'
```

```
## Warning in dmt(Qdata, df, rep(0, d), Sigma, log = TRUE): value out of range
## in 'lgamma'

## Warning in log(pi * df): NaNs produced

## Warning in nlminb(startdf, negloglik2, data = Udata, P = P, ...): NA/NaN
## function evaluation
```

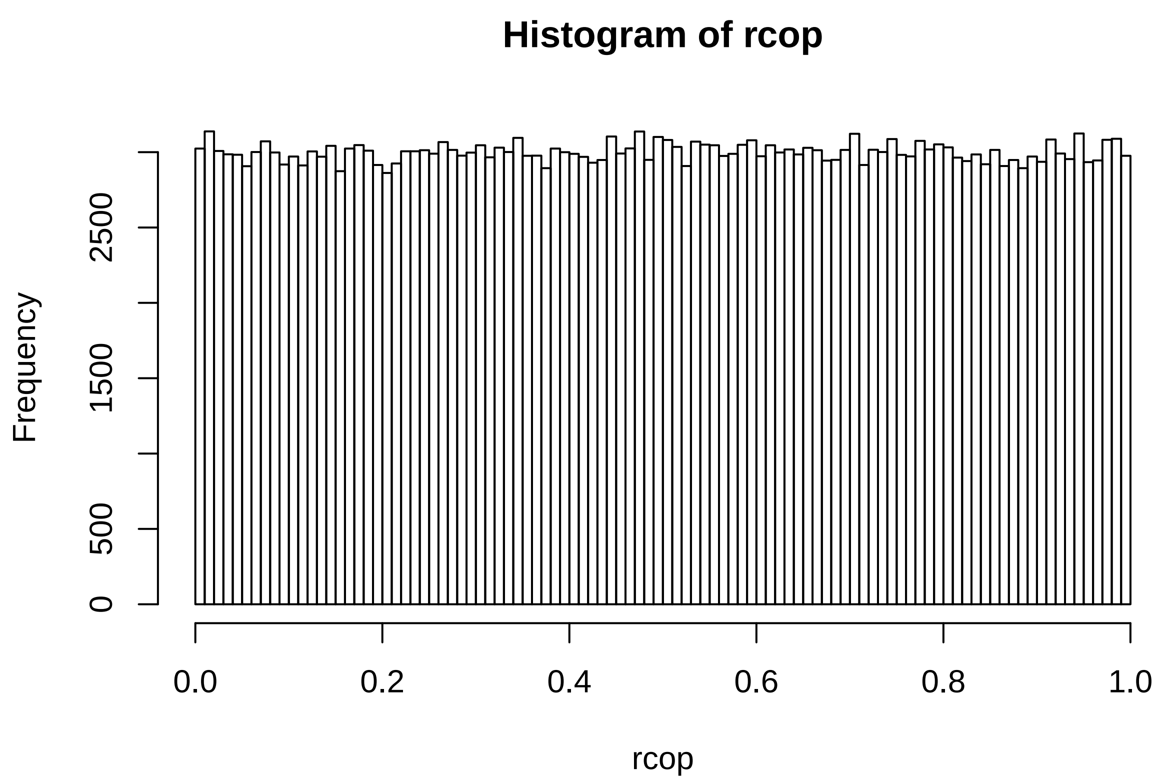
```
cop
```

```
## $P
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.4698119 0.2777477
## [2,] 0.4698119 1.0000000 0.4029480
## [3,] 0.2777477 0.4029480 1.0000000
##
## $nu
## [1] 0.7447845
##
## $converged
## [1] TRUE
##
## $ll.max
## [1] 413.7074
##
## $fit
## $fit$par
## [1] 0.7447845
##
## $fit$objective
## [1] -413.7074
##
## $fit$convergence
## [1] 0
##
## $fit$iterations
## [1] 8
##
## $fit$evaluations
## function gradient
##       13       11
##
## $fit$message
## [1] "relative convergence (4)"
```

```
### Step 5 : 100,000 random losses simulated for each financial instrument
rcop <- rcopula.t(100000, df = cop$nu, Sigma = cop$P)
head(rcop)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.5117571 0.4688324 0.6316129
## [2,] 0.6319909 0.5545535 0.6753459
## [3,] 0.8333424 0.2314036 0.6418810
## [4,] 0.6446113 0.3986271 0.4783977
## [5,] 0.3951009 0.2488283 0.2199989
## [6,] 0.9170151 0.9109124 0.9132352
```

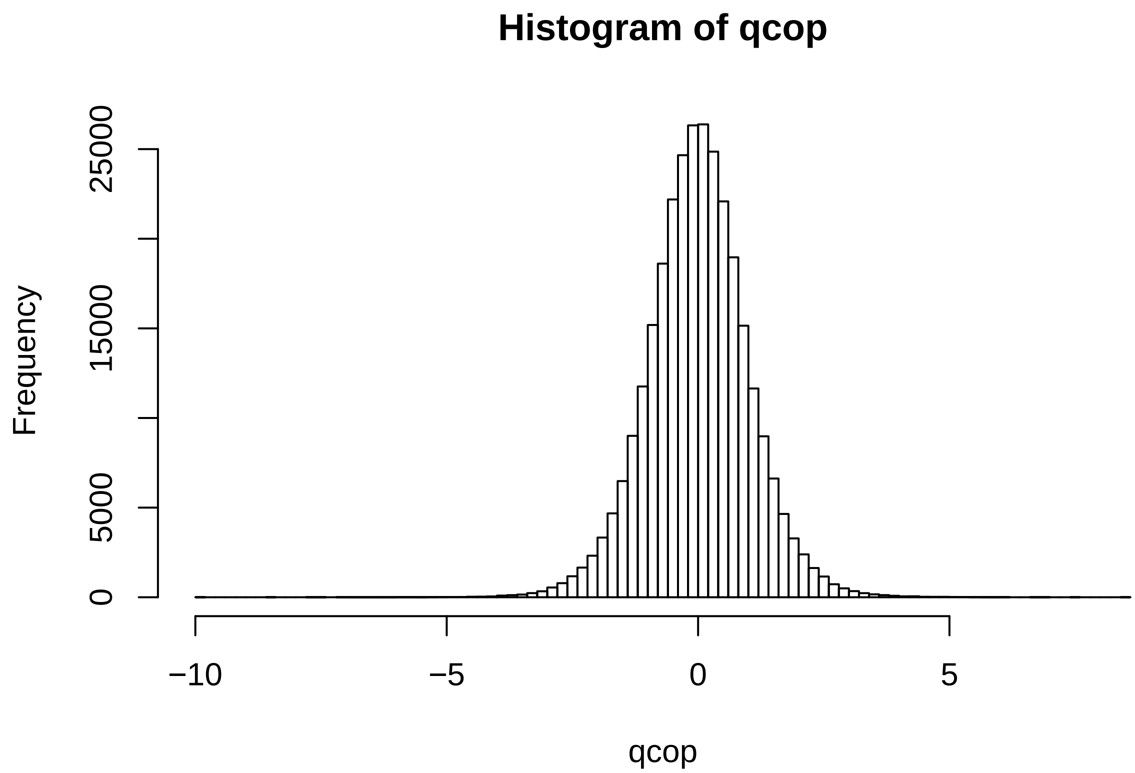
```r
hist(rcop, breaks = 100)
```




```r
### Step 6 : Compute the quantiles for these Monte Carlo draws.
qcop <- sapply(1:3, function(x) qstd(rcop[, x], nu = dshape[x]))
head(qcop)
```

```
##            [,1]        [,2]        [,3]
## [1,]  0.0263953 -0.07172815  0.30596161
## [2,]  0.3030086  0.12585498  0.41504042
## [3,]  0.8932077 -0.68288042  0.33105254
## [4,]  0.3335386 -0.23599313 -0.04915664
## [5,] -0.2387891 -0.62943944 -0.71284950
## [6,]  1.3214588  1.29532062  1.30387655
```

```r
hist(qcop, breaks = 100)
```




```
ht.mat <- matrix(dprog, nrow = 100000, ncol = ncol(dataloss), byrow = TRUE)
head(ht.mat)
```

```
##          [,1]     [,2]     [,3]
## [1,] 8.10235 10.08633 4.872721
## [2,] 8.10235 10.08633 4.872721
## [3,] 8.10235 10.08633 4.872721
## [4,] 8.10235 10.08633 4.872721
## [5,] 8.10235 10.08633 4.872721
## [6,] 8.10235 10.08633 4.872721
```

```
pf <- qcop * ht.mat
head(pf)
```

```
##           [,1]       [,2]       [,3]
## [1,]  0.213864 -0.7234741  1.4908656
## [2,]  2.455082  1.2694155  2.0223763
## [3,]  7.237082 -6.8877604  1.6131268
## [4,]  2.702446 -2.3803057 -0.2395266
## [5,] -1.934753 -6.3487368 -3.4735169
## [6,] 10.706922 13.0650371  6.3534270
```

```
## ES 95 percent
weights <- c(wIBM, wGE , wPG )    # from GMVP portfolio


### Step 7 : The simulated portfolio losses are then determined as the outcome of the matrix-weight vec
```

```r
pfall <- (qcop * ht.mat) %*% weights
head(pfall)
```

```
##            [,1]
## [1,]  0.8168185
## [2,]  2.0445734
## [3,]  2.0779526
## [4,]  0.3199598
## [5,] -3.4216827
## [6,]  8.5364917
```

```r
### Step 8
pfall.es95 <- median(tail(sort(pfall), 5000))
pfall.es95       # 10.3174
```

```
## [1] 10.37379
```

```r
pfall.var95 <- min(tail(sort(pfall), 5000))
pfall.var95     # 7.8486
```

```
## [1] 7.836884
```