Comparing Monte Carlo simulation with Topological Data Analysis- Part 2: Risk analysis

Miradain Atontsa (miradain.atontsan@gmail.com)

10/11/2020

The data

Loading the libraries

```
library(rmsfuns)
load_pkg(c("devtools", "rugarch", "forecast", "tidyr", "ggplot2", "parallel",
           "xtable", "zoo", "dplyr", "qrmtools", "tseries", "psych", "copula",
           "MASS", "crch", "metRology"))
#library(tseries)
```

Loading the data from Google finance

We will use the last 2000 days as the time period

```
# set dates
first.date <- Sys.Date() - 2000
last.date <- Sys.Date()
freq.data <- 'daily'
# set tickers
tickers <- c('GLE.PA', 'ML.PA', 'ENGI.PA', 'TEP.PA', 'EN.PA', 'BNP.PA',
             'VIE.PA', 'CA.PA', 'KER.PA', 'SU.PA')

l.out <- BatchGetSymbols::BatchGetSymbols(tickers = tickers,
                         first.date = first.date,
                         last.date = last.date,
                         freq.data = freq.data,
                         type.return = "arit",
                         cache.folder = file.path(tempdir(),
                                                  'BGS_Cache') ) # cache in tempdir()
# Extract data with only closing prices
df=l.out$df.tickers
features=c('ref.date', 'price.close', 'ticker')
df=df[features]
```

Data processing

```
# Fill missing values
df<-na.locf(df)

df_Kering<-df[ which(df$ticker =='KER.PA'),]
df_Schneider<-df[ which(df$ticker =='SU.PA'),]
```

```r
df_Carrefour<-df[ which(df$ticker =='CA.PA'),]
df_Vie<-df[ which(df$ticker =='VIE.PA'),]
df_BNP<-df[ which(df$ticker =='BNP.PA'),]
df_EN<-df[ which(df$ticker =='EN.PA'),]
df_Tep<-df[ which(df$ticker =='TEP.PA'),]
df_Engie<-df[ which(df$ticker =='ENGI.PA'),]
df_ML<-df[ which(df$ticker =='ML.PA'),]
df_GLE<-df[ which(df$ticker =='GLE.PA'),]




# Extracting close price data

data <- data_frame(date=df_Kering$ref.date,
                   GLE=df_GLE$price.close,
                   ML=df_ML$price.close,
                   Engie=df_Engie$price.close,
                   TEP=df_Tep$price.close,
                   EN=df_EN$price.close,
                   BNP=df_BNP$price.close,
                   Vie=df_Vie$price.close,
                   Carrefour=df_Carrefour$price.close,
                   Kering=df_Kering$price.close,
                   Schneider= df_Schneider$price.close,
                    )


# Compute the log return of close price data
data$Kering<-append( returns_qrmtools(data$Kering ,method='logarithmic'), 0, after = 0)
data$Schneider<-append( returns_qrmtools(data$Schneider ,method='logarithmic'), 0, after = 0)
data$Carrefour<-append( returns_qrmtools(data$Carrefour ,method='logarithmic'), 0, after = 0)
data$Vie<-append( returns_qrmtools(data$Vie ,method='logarithmic'), 0, after = 0)
data$BNP<-append( returns_qrmtools(data$BNP ,method='logarithmic'), 0, after = 0)
data$EN<-append( returns_qrmtools(data$EN ,method='logarithmic'), 0, after = 0)
data$TEP<-append( returns_qrmtools(data$TEP ,method='logarithmic'), 0, after = 0)
data$Engie<-append( returns_qrmtools(data$Engie ,method='logarithmic'), 0, after = 0)
data$ML<-append( returns_qrmtools(data$ML ,method='logarithmic'), 0, after = 0)
data$GLE<-append( returns_qrmtools(data$GLE ,method='logarithmic'), 0, after = 0)
```

- Save the data in external csv file

```r
write.table(x=data, file="Return_data.csv" , sep=";", dec = ",")
```

Portfolio weights by methods

This was done in part 1 for the Monte Carlo method and the weights for TDA was done in our first paper in this series ("Investing on CaC 40").

```r
weight.tda=c( 0.0000404, 0.000364, 0.002657, 0.0008564, 0.001257,
              0.000106, 0.0029905, 0.0021241, 0.95415, 0.035585)
weight.mc=c(0.002690, 0.053972,  0.047570, 0.199497, 0.014557,
            0.015709, 0.241335, 0.049635, 0.249791, 0.125243)
```

- Portfolio cummulative return

```
# Weights
weight<- data_frame(Assets=c('SG', 'Michelin', 'Engie', 'Teleperformance', 'Bouygues', 'BNP', 'Veolia',

# Portfolio return
portfolio_return<-cbind(data$GLE, data$ML, data$Engie, data$TEP,
                        data$EN, data$BNP, data$Vie, data$Carrefour,
                        data$Kering, data$Schneider)

return.tda<-as.numeric(as.matrix(portfolio_return) %*% weight.tda)
return.mc<-as.numeric(as.matrix(portfolio_return) %*% weight.mc)

# Cummulative return
Cummulative_return_tda<- cumprod(1+return.tda)
Cummulative_return_mc<- cumprod(1+return.mc)
Cummulative_return<-data_frame(date=data$date, return.tda=Cummulative_return_tda,
                               return.mc= Cummulative_return_mc)
```

- Save the data in external csv file

```
write.table(x=weight, file="weight.csv", sep=";", dec = ",") # Save weights for assets

write.table(x=Cummulative_return, file="Cummulative_return.csv" , sep=";", dec = ",")
# Save cummulative returns
```

Fitting univariate GARCH (using rugarch)

Case of Kering

- Mean model selection

```
# Kering case
cl=makePSOCKcluster(10)
AC_K= autoarfima(as.numeric(data$Kering), ar.max = 2, ma.max = 2,
                 criterion = "AIC", method = 'partial')
show(head(AC_K$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  2    1      0 -5.015540         1
## 2  2  0    1      0 -5.015390         1
## 3  1  0    1      0 -5.015233         1
## 4  0  1    1      0 -5.015079         1
## 5  0  2    0      0 -5.014682         1
## 6  0  0    1      0 -5.014617         1
```

- Fitting the ARMA(0,2)-GARCH(1,1) (with t innovation) model

```
# Note we specify the mean (m) and variance (sigma) models separately
garch.K<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(0,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
```

```r
garchfit.K=ugarchfit(spec = garch.K, data=as.numeric(data$Kering))

# Table of the model parameters
Table.K<-garchfit.K@fit$matcoef
Table.K
```

```
##              Estimate   Std. Error   t value    Pr(>|t|)
## mu      1.301897e-03 3.553720e-04  3.663476 0.000248816
## ma1    -4.776529e-02 2.510555e-02 -1.902579 0.057095461
## ma2    -3.416005e-02 2.608667e-02 -1.309483 0.190370855
## omega   7.620911e-06 4.634496e-06  1.644388 0.100096056
## alpha1  6.092161e-02 7.268523e-03  8.381566 0.000000000
## beta1   9.245539e-01 9.634736e-03 95.960477 0.000000000
## shape   3.773596e+00 2.793957e-01 13.506279 0.000000000
```

```r
alpha<-0.05
# Simulation
sim.K<-ugarchsim(garchfit.K, n.sim = 10000, rseed = 13) # Simulation on the ARMA-GARCH model
r.K<-fitted(sim.K) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.K<-sim.K@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.K<-quantile(r.K, alpha) # VaR
ES.K<-mean(r.K[r.K<VaR.K])  # ES
round(VaR.K, 6)
```

```
##        5%
## -0.030797
```

```r
round(ES.K, 6)
```

```
## [1] -0.058309
```

Case of Schneider

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_K= autoarfima(as.numeric(data$Schneider), ar.max = 2, ma.max = 2,
                 criterion = "AIC", method = 'partial')
show(head(AC_K$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  0    1      0 -5.288794         1
## 2  1  2    0      0 -5.288680         1
## 3  0  1    0      0 -5.288437         1
## 4  1  0    0      0 -5.288398         1
## 5  1  2    1      0 -5.287962         1
## 6  2  1    1      0 -5.287959         1
```

- Fitting the ARMA(0,0)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.S<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
```

```r
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(0,0), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.S=ugarchfit(spec = garch.S, data=as.numeric(data$Schneider))

# Table of the model parameters
Table<-garchfit.S@fit$matcoef
Table
```

```
##             Estimate   Std. Error   t value     Pr(>|t|)
## mu     8.648346e-04 3.384831e-04  2.555030 1.061785e-02
## omega  6.558176e-06 5.054363e-06  1.297528 1.944497e-01
## alpha1 7.413726e-02 1.087493e-02  6.817261 9.279244e-12
## beta1  9.053338e-01 1.339262e-02 67.599472 0.000000e+00
## shape  4.661664e+00 4.907487e-01  9.499085 0.000000e+00
```

```r
# Simulation
sim.S<-ugarchsim(garchfit.S, n.sim = 10000, rseed = 14) # Simulation on the ARMA-GARCH model
r.S<-fitted(sim.S) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.S<-sim.S@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.S<-quantile(r.S, alpha) # VaR
ES.S<-mean(r.S[r.S<VaR.S])  # ES
round(VaR.S, 6)
```

```
##        5%
## -0.023967
```

```r
round(ES.S, 6)
```

```
## [1] -0.037233
```

Case of Carrefour

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_ca= autoarfima(as.numeric(data$Carrefour), ar.max = 2, ma.max = 2,
                  criterion = "AIC", method = 'partial')
show(head(AC_ca$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  0    1      0 -5.269961         1
## 2  2  2    0      0 -5.269338         1
## 3  2  2    1      0 -5.269015         1
## 4  0  2    0      0 -5.268943         1
## 5  0  1    0      0 -5.268820         1
## 6  2  0    0      0 -5.268781         1
```

- Fitting the ARMA(0,0)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately
```

```r
garch.ca<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(0,0), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.ca=ugarchfit(spec = garch.ca, data=as.numeric(data$Carrefour))

# Table of the model parameters
Table<-garchfit.ca@fit$matcoef
Table
```

```
##             Estimate   Std. Error    t value     Pr(>|t|)
## mu     -3.585910e-04 3.370160e-04  -1.064018 0.2873206415
## omega   1.747823e-06 4.866440e-07   3.591584 0.0003286745
## alpha1  2.312514e-02 1.778251e-03  13.004424 0.0000000000
## beta1   9.732817e-01 2.270428e-03 428.677706 0.0000000000
## shape   3.375483e+00 3.463601e-01   9.745588 0.0000000000
```

```r
# Simulation
sim.ca<-ugarchsim(garchfit.ca, n.sim = 10000, rseed = 15) # Simulation on the ARMA-GARCH model
r.ca<-fitted(sim.ca) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.ca<-sim.ca@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.ca<-quantile(r.ca, alpha) # VaR
ES.ca<-mean(r.ca[r.ca<VaR.ca])  # ES
round(VaR.ca, 6)
```

```
##        5%
## -0.039801
```

```r
round(ES.ca, 6)
```

```
## [1] -0.072104
```

Case of Veolia (Vie)

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_vie= autoarfima(as.numeric(data$Vie), ar.max = 2, ma.max = 2,
                 criterion = "AIC", method = 'partial')
show(head(AC_vie$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  2    0      0 -5.418633         1
## 2  2  0    0      0 -5.418556         1
## 3  1  0    0      0 -5.418055         1
## 4  0  1    0      0 -5.417940         1
## 5  1  1    0      0 -5.417561         1
## 6  1  2    0      0 -5.417327         1
```

- Fitting the ARMA(2,2)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.vie<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(2,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.vie=ugarchfit(spec = garch.vie, data=as.numeric(data$Vie))

# Table of the model parameters
Table<-garchfit.vie@fit$matcoef
Table
```

```
##              Estimate   Std. Error       t value     Pr(>|t|)
## mu      8.593957e-04 2.583186e-04  3.326882e+00 8.782347e-04
## ar1     1.846893e+00 6.742130e-03  2.739332e+02 0.000000e+00
## ar2    -9.189318e-01 6.607517e-03 -1.390737e+02 0.000000e+00
## ma1    -1.873003e+00 7.030302e-06 -2.664185e+05 0.000000e+00
## ma2     9.374798e-01 1.852159e-04  5.061550e+03 0.000000e+00
## omega   3.232644e-06 8.462883e-06  3.819790e-01 7.024769e-01
## alpha1  7.194153e-02 6.029165e-02  1.193226e+00 2.327811e-01
## beta1   9.208474e-01 6.267938e-02  1.469139e+01 0.000000e+00
## shape   3.849720e+00 6.273949e-01  6.136040e+00 8.460410e-10
```

```r
# Simulation
sim.vie<-ugarchsim(garchfit.vie, n.sim = 10000, rseed = 16) # Simulation on the ARMA-GARCH model
r.vie<-fitted(sim.vie) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.vie<-sim.vie@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.vie<-quantile(r.vie, alpha) # VaR
ES.vie<-mean(r.vie[r.vie<VaR.vie])  # ES
round(VaR.vie, 6)
```

```
##        5%
## -0.023772
```

```r
round(ES.vie, 6)
```

```
## [1] -0.044624
```

Case of BNP

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_bnp= autoarfima(as.numeric(data$BNP), ar.max = 2, ma.max = 2,
                  criterion = "AIC", method = 'partial')
show(head(AC_bnp$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  2  2    0      0 -4.898183         1
## 2  2  2    1      0 -4.897088         1
```

```
## 3  1  0    0      0 -4.890358         1
## 4  0  1    0      0 -4.890239         1
## 5  0  2    0      0 -4.890107         1
## 6  2  0    0      0 -4.889907         1
```

- Fitting the ARMA(2,2)-GARCH(1,1) (with t innovation) model

```
# Note we specify the mean (m) and variance (sigma) models separately

garch.bnp<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(2,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.bnp=ugarchfit(spec = garch.bnp, data=as.numeric(data$BNP))

# Table of the model parameters
Table<-garchfit.bnp@fit$matcoef
Table
```

```
##             Estimate   Std. Error    t value     Pr(>|t|)
## mu     -1.531760e-04 3.954608e-04 -0.3873353 6.985080e-01
## ar1     8.142889e-01 2.785483e-01  2.9233308 3.463084e-03
## ar2    -6.043676e-01 1.763148e-01 -3.4277755 6.085485e-04
## ma1    -7.342129e-01 2.903276e-01 -2.5289116 1.144169e-02
## ma2     5.506904e-01 1.759550e-01  3.1297229 1.749712e-03
## omega   5.746707e-06 3.621856e-06  1.5866747 1.125863e-01
## alpha1  7.921393e-02 1.630129e-02  4.8593661 1.177622e-06
## beta1   9.086697e-01 1.757291e-02 51.7085577 0.000000e+00
## shape   5.272383e+00 7.406642e-01  7.1184517 1.091571e-12
```

```
# Simulation
sim.bnp<-ugarchsim(garchfit.bnp, n.sim = 10000, rseed = 17) # Simulation on the ARMA-GARCH model
r.bnp<-fitted(sim.bnp) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.bnp<-sim.bnp@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.bnp<-quantile(r.bnp, alpha) # VaR
ES.bnp<-mean(r.bnp[r.bnp<VaR.bnp])  # ES
round(VaR.bnp, 6)
```

```
##        5%
## -0.030209
```

```
round(ES.bnp, 6)
```

```
## [1] -0.046222
```

Case of Bouygues (EN)

- Mean model selection

```
# Schneider case
cl=makePSOCKcluster(10)
AC_EN= autoarfima(as.numeric(data$EN), ar.max = 2, ma.max = 2,
```

```r
                 criterion = "AIC", method = 'partial')
show(head(AC_EN$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  2  2    0      0 -5.045139         1
## 2  2  2    1      0 -5.043761         1
## 3  0  2    0      0 -5.037344         1
## 4  2  0    0      0 -5.037236         1
## 5  1  0    0      0 -5.036540         1
## 6  0  1    0      0 -5.036491         1
```

- Fitting the ARMA(2,2)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.EN<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                      garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(2,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.EN=ugarchfit(spec = garch.EN, data=as.numeric(data$EN))

# Table of the model parameters
Table<-garchfit.EN@fit$matcoef
Table
```

```
##             Estimate   Std. Error       t value     Pr(>|t|)
## mu      2.992953e-04 3.235022e-04     0.9251724 3.548762e-01
## ar1     3.394745e-01 4.954971e-03    68.5119021 0.000000e+00
## ar2    -9.919763e-01 2.783340e-03  -356.3978713 0.000000e+00
## ma1    -3.470873e-01 4.486331e-03   -77.3655137 0.000000e+00
## ma2     9.958938e-01 6.327439e-05 15739.2884008 0.000000e+00
## omega   1.403829e-05 2.561257e-06     5.4810150 4.228927e-08
## alpha1  1.155914e-01 7.233202e-03    15.9806725 0.000000e+00
## beta1   8.415332e-01 1.853188e-02    45.4100262 0.000000e+00
## shape   4.060559e+00 3.456767e-01    11.7466947 0.000000e+00
```

```r
# Simulation
sim.EN<-ugarchsim(garchfit.EN, n.sim = 10000, rseed = 18) # Simulation on the ARMA-GARCH model
r.EN<-fitted(sim.EN) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.EN<-sim.EN@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.EN<-quantile(r.EN, alpha) # VaR
ES.EN<-mean(r.EN[r.EN<VaR.EN])  # ES
round(VaR.EN, 6)
```

```
##        5%
## -0.025374
```

```r
round(ES.EN, 6)
```

```
## [1] -0.039207
```

Case of Teleperformance (TEP)

- Mean model selection

```
# Schneider case
cl=makePSOCKcluster(10)
AC_tep= autoarfima(as.numeric(data$TEP), ar.max = 2, ma.max = 2,
                  criterion = "AIC", method = 'partial')
show(head(AC_tep$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  2  2    1      0 -5.272109         1
## 2  1  0    1      0 -5.267181         1
## 3  0  1    1      0 -5.266941         1
## 4  1  1    1      0 -5.266391         1
## 5  2  0    1      0 -5.266128         1
## 6  0  2    1      0 -5.265924         1
```

- Fitting the ARMA(2,2)-GARCH(1,1) (with t innovation) model

```
# Note we specify the mean (m) and variance (sigma) models separately

garch.tep<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                      garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(2,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.tep=ugarchfit(spec = garch.tep, data=as.numeric(data$TEP))

# Table of the model parameters
Table<-garchfit.tep@fit$matcoef
Table
```

```
##            Estimate   Std. Error    t value     Pr(>|t|)
## mu      1.149802e-03 2.215120e-04  5.1906980 2.095072e-07
## ar1     4.427521e-02 3.314813e-01  0.1335677 8.937444e-01
## ar2     6.866564e-01 2.634665e-01  2.6062380 9.154282e-03
## ma1    -1.063622e-01 3.379622e-01 -0.3147162 7.529771e-01
## ma2    -7.111319e-01 2.900006e-01 -2.4521743 1.419959e-02
## omega   1.910839e-05 7.378729e-06  2.5896589 9.607108e-03
## alpha1  1.630268e-01 5.512265e-02  2.9575288 3.101158e-03
## beta1   7.756593e-01 6.745188e-02 11.4994462 0.000000e+00
## shape   4.396608e+00 5.056520e-01  8.6949281 0.000000e+00
```

```
# Simulation
sim.tep<-ugarchsim(garchfit.tep, n.sim = 10000, rseed = 19) # Simulation on the ARMA-GARCH model
r.tep<-fitted(sim.tep) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.tep<-sim.tep@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.tep<-quantile(r.tep, alpha) # VaR
ES.tep<-mean(r.tep[r.tep<VaR.tep])  # ES
round(VaR.tep, 6)
```

```
##        5%
```

```
## -0.022065
```

```r
round(ES.tep, 6)
```

```
## [1] -0.038984
```

Case of Engie

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_engie= autoarfima(as.numeric(data$Engie), ar.max = 2, ma.max = 2,
                 criterion = "AIC", method = 'partial')
show(head(AC_engie$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  2  2    0      0 -5.371065         1
## 2  2  2    1      0 -5.369989         1
## 3  1  0    0      0 -5.368445         1
## 4  0  1    0      0 -5.368354         1
## 5  1  0    1      0 -5.367335         1
## 6  0  1    1      0 -5.367247         1
```

- Fitting the ARMA(1,0)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.engie<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(1,0), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.engie=ugarchfit(spec = garch.engie, data=as.numeric(data$Engie))

# Table of the model parameters
Table<-garchfit.engie@fit$matcoef
Table
```

```
##            Estimate   Std. Error    t value    Pr(>|t|)
## mu     1.620484e-04 3.281668e-04  0.4937987 0.621448325
## ar1    3.019046e-02 2.572216e-02  1.1737141 0.240509546
## omega  6.972980e-06 2.153075e-06  3.2386146 0.001201118
## alpha1 6.888237e-02 5.707014e-03 12.0697731 0.000000000
## beta1  9.051868e-01 1.482753e-02 61.0477299 0.000000000
## shape  4.275690e+00 4.498320e-01  9.5050812 0.000000000
```

```r
# Simulation
sim.engie<-ugarchsim(garchfit.engie, n.sim = 10000, rseed = 20) # Simulation on the ARMA-GARCH model
r.engie<-fitted(sim.engie) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.engie<-sim.engie@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.engie<-quantile(r.engie, alpha) # VaR
```

```r
ES.engie<-mean(r.engie[r.engie<VaR.engie])  # ES
round(VaR.engie, 6)
```

```
##       5%
## -0.02389
```

```r
round(ES.engie, 6)
```

```
## [1] -0.039532
```

Case of Michelin (ML)

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_ml= autoarfima(as.numeric(data$ML), ar.max = 2, ma.max = 2,
                  criterion = "AIC", method = 'partial')
show(head(AC_ml$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  1    0      0 -5.306456         1
## 2  1  0    0      0 -5.306442         1
## 3  0  0    1      0 -5.305968         1
## 4  2  2    0      0 -5.305768         1
## 5  0  2    0      0 -5.305241         1
## 6  2  0    0      0 -5.305181         1
```

- Fitting the ARMA(0,1)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.ml<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(0,1), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.ml=ugarchfit(spec = garch.ml, data=as.numeric(data$ML))

# Table of the model parameters
Table<-garchfit.ml@fit$matcoef
Table
```

```
##             Estimate   Std. Error    t value     Pr(>|t|)
## mu      2.480136e-04 3.497086e-04  0.7092007 4.781999e-01
## ma1    -2.020617e-02 2.664663e-02 -0.7583010 4.482708e-01
## omega   3.919796e-06 2.455064e-06  1.5966169 1.103511e-01
## alpha1  4.073049e-02 1.022964e-02  3.9816150 6.844859e-05
## beta1   9.432714e-01 1.160630e-02 81.2723981 0.000000e+00
## shape   5.812273e+00 8.301480e-01  7.0014902 2.532641e-12
```

```r
# Simulation
sim.ml<-ugarchsim(garchfit.ml, n.sim = 10000, rseed = 21) # Simulation on the ARMA-GARCH model
r.ml<-fitted(sim.ml) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
```

```r
sim.sigma.ml<-sim.ml@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.ml<-quantile(r.ml, alpha) # VaR
ES.ml<-mean(r.ml[r.ml<VaR.ml])  # ES
round(VaR.ml, 6)
```

```
##        5%
## -0.023556
```

```r
round(ES.ml, 6)
```

```
## [1] -0.033722
```

Case of Société Générale (GLE)

- Mean model selection

```r
# Schneider case
cl=makePSOCKcluster(10)
AC_gle= autoarfima(as.numeric(data$GLE), ar.max = 2, ma.max = 2,
                  criterion = "AIC", method = 'partial')
show(head(AC_gle$rank.matrix))
```

```
##   AR MA Mean ARFIMA       AIC converged
## 1  2  2    0      0 -4.532790         1
## 2  2  2    1      0 -4.532648         1
## 3  0  2    0      0 -4.530621         1
## 4  0  2    1      0 -4.530374         1
## 5  2  0    0      0 -4.530300         1
## 6  2  0    1      0 -4.530059         1
```

- Fitting the ARMA(2,2)-GARCH(1,1) (with t innovation) model

```r
# Note we specify the mean (m) and variance (sigma) models separately

garch.gle<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "fGARCH", "apARCH")[1],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(2,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[3]
  )
# Now fit
garchfit.gle=ugarchfit(spec = garch.gle, data=as.numeric(data$GLE))

# Table of the model parameters
Table<-garchfit.gle@fit$matcoef
Table
```

```
##            Estimate   Std. Error     t value     Pr(>|t|)
## mu    -2.893316e-04 4.117196e-04  -0.7027394 4.822182e-01
## ar1    1.090732e+00 6.011081e-02  18.1453505 0.000000e+00
## ar2   -9.093229e-01 6.424663e-02 -14.1536268 0.000000e+00
## ma1   -1.052376e+00 7.188975e-02 -14.6387433 0.000000e+00
## ma2    8.772643e-01 7.347155e-02  11.9401910 0.000000e+00
## omega  6.160905e-06 3.768749e-06   1.6347346 1.021047e-01
```

```
## alpha1  8.457491e-02 1.780340e-02   4.7504928 2.029216e-06
## beta1   9.096682e-01 1.786513e-02  50.9186500 0.000000e+00
## shape   4.414116e+00 5.345368e-01   8.2578336 2.220446e-16
```

```
# Simulation
sim.gle<-ugarchsim(garchfit.gle, n.sim = 10000, rseed = 22) # Simulation on the ARMA-GARCH model
r.gle<-fitted(sim.gle) # simulated process r_t=mu_t + w_t for w_t= sigma_t * z_t
sim.sigma.gle<-sim.gle@simulation$sigmaSim # GARCH sigma simulation

# Risk measurement
VaR.gle<-quantile(r.gle, alpha) # VaR
ES.gle<-mean(r.gle[r.gle<VaR.gle])  # ES
round(VaR.gle, 6)
```

```
##        5%
## -0.038472
```

```
round(ES.gle, 6)
```

```
## [1] -0.063583
```

VaR by assets

```
# Non-parametric VaR
VaR.assets<-data_frame(Assets=c('Kering', 'Schneider', 'Carrefour', 'Veolia', 'BNP',
                                'Bouygues','Teleperformance', 'Engie', 'Michelin', 'SG' ),
                       VaR=c(VaR.K,VaR.S,VaR.ca, VaR.vie, VaR.bnp, VaR.EN, VaR.tep,
                             VaR.engie, VaR.ml,VaR.gle),
                       Expected_Shortfall= c(ES.K,ES.S,ES.ca, ES.vie, ES.bnp, ES.EN,
                                             ES.tep, ES.engie, ES.ml,ES.gle))
```

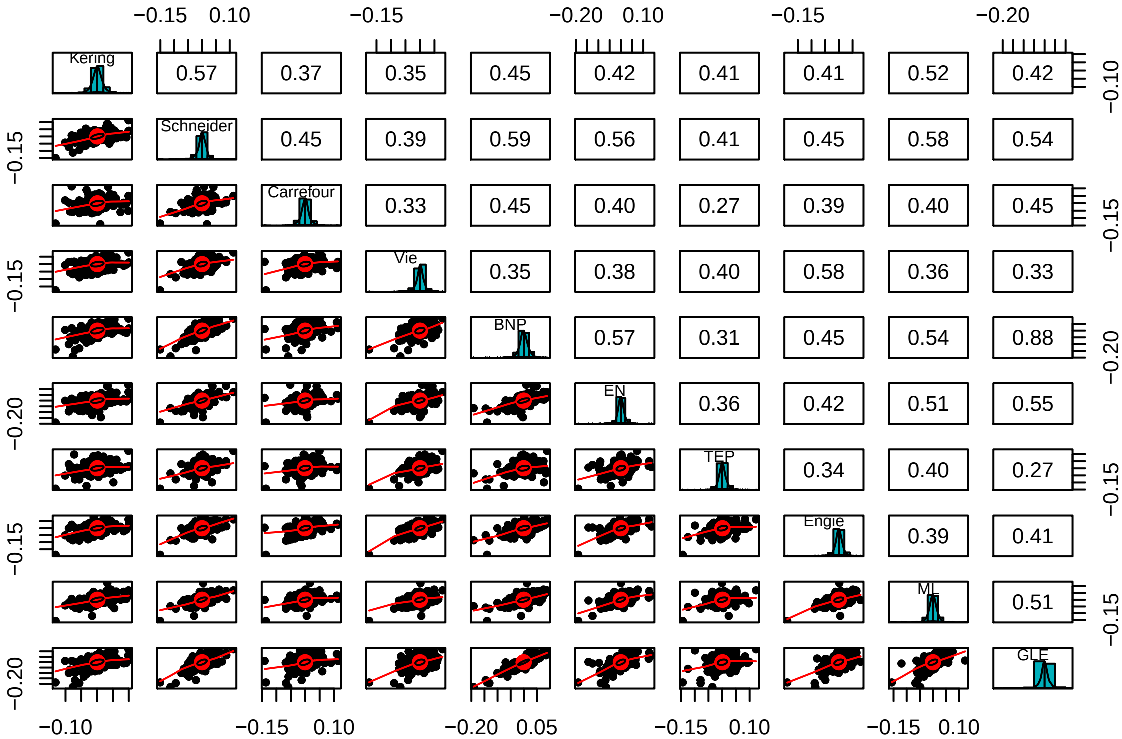
- Save the data in external csv file

```
write.table(x=VaR.assets, file="VaR.assets.csv" , sep=";", dec = ",") # Save VaR for assets
```

Fitting multivariate GARCH

Plotting the distributions

```
pairs.panels(data[c('Kering','Schneider','Carrefour','Vie','BNP','EN','TEP','Engie','ML','GLE')],
             method = "spearman", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```




```
#'Kering','Schneider','Carrefour','Vie','BNP','EN','TEP','Engie','ML','GLE'
```

Model residual

```
wt.K<-residuals(garchfit.K)    # Ordinary resids
sigma.K<-sigma(garchfit.K)     # Conditional resids
zt.K<- residuals( garchfit.K, standardize=TRUE)   # Standardized resids


wt.S<-residuals(garchfit.S)    # Ordinary resids
sigma.S<-sigma(garchfit.S)     # Conditional resids
zt.S<- residuals( garchfit.S, standardize=TRUE)   # Standardized resids

wt.ca<-residuals(garchfit.ca)    # Ordinary resids
sigma.ca<-sigma(garchfit.ca)     # Conditional resids
zt.ca<- residuals( garchfit.ca, standardize=TRUE)   # Standardized resids

wt.EN<-residuals(garchfit.EN)    # Ordinary resids
sigma.EN<-sigma(garchfit.EN)     # Conditional resids
zt.EN<- residuals( garchfit.EN, standardize=TRUE)   # Standardized resids

wt.engie<-residuals(garchfit.engie)    # Ordinary resids
sigma.engie<-sigma(garchfit.engie)     # Conditional resids
zt.engie<- residuals( garchfit.engie, standardize=TRUE)   # Standardized resids


wt.gle<-residuals(garchfit.gle)    # Ordinary resids
sigma.gle<-sigma(garchfit.gle)     # Conditional resids
zt.gle<- residuals( garchfit.gle, standardize=TRUE)   # Standardized resids
```

```
wt.ml<-residuals(garchfit.ml)   # Ordinary resids
sigma.ml<-sigma(garchfit.ml)    # Conditional resids
zt.ml<- residuals( garchfit.ml, standardize=TRUE)  # Standardized resids

wt.tep<-residuals(garchfit.tep)   # Ordinary resids
sigma.tep<-sigma(garchfit.tep)    # Conditional resids
zt.tep<- residuals( garchfit.tep, standardize=TRUE)  # Standardized resids

wt.vie<-residuals(garchfit.vie)   # Ordinary resids
sigma.vie<-sigma(garchfit.vie)    # Conditional resids
zt.vie<- residuals( garchfit.vie, standardize=TRUE)  # Standardized resids

wt.bnp<-residuals(garchfit.bnp)   # Ordinary resids
sigma.bnp<-sigma(garchfit.bnp)    # Conditional resids
zt.bnp<- residuals( garchfit.bnp, standardize=TRUE)  # Standardized resids
```

t-distribution functions

The asset returns fit the most with the student t distribution which is more appropriate for fat tail distributions. We define here below a couple of functions that we will sometimes use in our analysis.

```
dct<-function(x,m,s,df){return(dt((x-m)/s, df)/s)}
qct<-function(p,m,s,df){return(m+s*qt(p,df))}
pct<-function(q,m,s,df){return(pt((q-m)/s,df))}

uRes_data<-data_frame(uKering=pobs(zt.K), uSchneider=pobs(zt.S))
```

Copulas

Fit a t-copula

- Corelation

```
uRes_data<-data_frame(uKering=pobs(zt.K), uSchneider=pobs(zt.S), uCarrefour=pobs(zt.ca),
                      uVie=pobs(zt.vie),uBNP=pobs(zt.bnp), uEN=pobs(zt.EN),
                      uTEP=pobs(zt.tep), uEngie=pobs(zt.engie),
                      uML=pobs(zt.ml), uGLE=pobs(zt.gle))

correlation=cor(uRes_data, method = 'spearman')
```

- Gaussian copulas

```
param.cor<-correlation[upper.tri(correlation)]
# Creating the normal copula object
normal_Cop <- normalCopula(param=param.cor, dim = 10, dispstr = "un")

# fit Gaussian copula: estimating rho along the way
fit.gaussian<-fitCopula(normal_Cop, uRes_data, method="ml",
                        start=rep(mean(param.cor), 45), optim.control=list(maxit=2000) )
```

- Student t copulas

```r
dist_param<-apply (data[c('Kering','Schneider','Carrefour','Vie','BNP','EN','TEP','Engie',
                         'ML','GLE')], 2, fitdistr, "t")

# We consider the degree of freedom as the mean of the degree of freedom of the marginal distributions
start_df=mean(c(dist_param$Kering$estimate[3], dist_param$Schneider$estimate[3],
                dist_param$Carrefour$estimate[3], dist_param$Vie$estimate[3],
                dist_param$BNP$estimate[3],dist_param$EN$estimate[3],
                dist_param$TEP$estimate[3], dist_param$Engie$estimate[3],
                dist_param$ML$estimate[3],dist_param$GLE$estimate[3]))

# Creating the t-copula object
tcop<- tCopula(param = correlation[upper.tri(correlation)], dim=10, dispstr = "un")

# Fitting the t-copula:
fit.t<-fitCopula(tcop, uRes_data, method="ml",
                 start= c(fit.gaussian@estimate, df=start_df),
                 optim.control=list(maxit=2000))

# Record the AIC of the fit
#fit.t.aic<- -2*fit.t@loglik + 2*length(fit.t@estimate)
```

Monte Carlo simulation on residuals

We will now generate 10 000 marginal resid values for the Kering and Schneider stock. We will next use the simulated GARCH(1,1) volatility values to recover simulated stock returns. We will then compute the non-parametric VaR and ES on the obtained values.

Copula distribution

```r
copula_distribution<- mvdc(copula = tcop, margins = c("ct","ct", "ct","ct", "ct","ct", "ct","ct", "ct",
  paramMargins = list(
        list(m=dist_param$Kering$estimate[1], s=dist_param$Kering$estimate[2],
             df=dist_param$Kering$estimate[3]),
        list(m=dist_param$Schneider$estimate[1], s=dist_param$Schneider$estimate[2],
             df=dist_param$Schneider$estimate[3]),
        list(m=dist_param$Carrefour$estimate[1], s=dist_param$Carrefour$estimate[2],
             df=dist_param$Carrefour$estimate[3]),
        list(m=dist_param$Vie$estimate[1], s=dist_param$Vie$estimate[2],
             df=dist_param$Vie$estimate[3]),
        list(m=dist_param$BNP$estimate[1], s=dist_param$BNP$estimate[2],
             df=dist_param$BNP$estimate[3] ),
        list(m=dist_param$EN$estimate[1], s=dist_param$EN$estimate[2],
             df=dist_param$EN$estimate[3]),
        list(m=dist_param$TEP$estimate[1], s=dist_param$TEP$estimate[2],
             df=dist_param$TEP$estimate[3]),
        list(m=dist_param$Engie$estimate[1], s=dist_param$Engie$estimate[2],
             df=dist_param$Engie$estimate[3]),
        list(m=dist_param$ML$estimate[1], s=dist_param$ML$estimate[2],
             df=dist_param$ML$estimate[3]),
        list(m=dist_param$GLE$estimate[1], s=dist_param$GLE$estimate[2],
             df=dist_param$GLE$estimate[3])))
```

Simulation

```
# Simulate copula distribution values
set.seed(1)
sim<-rMvdc(10000, mvdc =copula_distribution) # Simulate marginal values for the standardized resids of
```

- Kering

```
# Function to compute the simulated Kering returns
funcSim.K<-function(copSim){
  serie.sim.K<-rep(0, 10000)
  w.t<-copSim * sim.sigma.K
  serie.sim.K[1]=1.267* 10^{-3}+w.t[1]
  serie.sim.K[2]=1.267* 10^{-3}+w.t[2]-0.0454*w.t[1]
  for (i in 3:10000) {
    serie.sim.K[i]=1.267* 10^{-3}+w.t[i]-0.0454*w.t[i-1]-0.0324 * w.t[i-2]
  }
  return(serie.sim.K)
}

# Simulated returns per stock
serie.sim.K=funcSim.K(sim[,1])   # Simulated series of Kering
```

- Schneider

```
# Function to compute the simulated Schneider returns
funcSim.S<-function(copSim){
  serie.sim.S<-rep(0, 10000)
  w.t<- copSim * sim.sigma.S
  for (i in 1:10000) {
    serie.sim.S[i]=8.44* 10^{-4}+ w.t[i]
  }
  return(serie.sim.S)
}


# Simulated returns per stock
serie.sim.S=funcSim.S(sim[,2])  # Simulated series of Schneider
```

- Carrefour

```
# Function to compute the simulated Schneider returns
funcSim.ca<-function(copSim){
  serie.sim.ca<-rep(0, 10000)
  w.t<- copSim * sim.sigma.ca
  for (i in 1:10000) {
    serie.sim.ca[i]=-3.67* 10^{-4}+ w.t[i]
  }
  return(serie.sim.ca)
}


# Simulated returns per stock
serie.sim.ca=funcSim.ca(sim[,3])  # Simulated series of Schneider
```

- Veolia (Vie)

```r
# Function to compute the simulated Schneider returns
funcSim.vie<-function(copSim){
  serie.sim.vie<-rep(0, 10000)
  w.t<- copSim * sim.sigma.vie
  serie.sim.vie[1]=8.54* 10^{-4}+w.t[1]
  serie.sim.vie[2]=8.54* 10^{-4}+ 1.85*(serie.sim.vie[1]-8.54* 10^{-4})+ w.t[2]-1.87*w.t[1]
  for (i in 3:10000) {
    serie.sim.vie[i]=8.54* 10^{-4}+ 1.85*(serie.sim.vie[i-1]-8.54* 10^{-4})-
      0.92*(serie.sim.vie[i-2]-8.54* 10^{-4})+ w.t[i]-1.87*w.t[i-1]+0.94 * w.t[i-2]
  }
  return(serie.sim.vie)
}


# Simulated returns per stock
serie.sim.vie=funcSim.vie(sim[,4])  # Simulated series of VIE
```

- BNP

```r
# Function to compute the simulated Schneider returns
funcSim.bnp<-function(copSim){
  serie.sim.bnp<-rep(0, 10000)
  w.t<- copSim * sim.sigma.bnp
  serie.sim.bnp[1]=-1.41* 10^{-4}+w.t[1]
  serie.sim.bnp[2]=-1.41* 10^{-4}+ 0.81*(serie.sim.bnp[1]+1.41* 10^{-4})+ w.t[2]-0.733*w.t[1]
  for (i in 3:10000) {
    serie.sim.bnp[i]=-1.41* 10^{-4}+ 0.81*(serie.sim.bnp[i-1]+1.41*
                                         10^{-4})-0.6*(serie.sim.bnp[i-2]+1.41*
                                                         10^{-4})+w.t[i]-0.733*w.t[i-1]+0.55 * w.t[i
  }
  return(serie.sim.bnp)
}


# Simulated returns per stock
serie.sim.bnp=funcSim.bnp(sim[,5])  # Simulated series of BNP
```

- Bouygues (EN)

```r
# Function to compute the simulated Schneider returns
funcSim.en<-function(copSim){
  serie.sim.en<-rep(0, 10000)
  w.t<- copSim * sim.sigma.EN
  serie.sim.en[1]=3.05* 10^{-4}+w.t[1]
  serie.sim.en[2]=3.05* 10^{-4}+ 0.445*(serie.sim.en[1]-3.05* 10^{-4})+ w.t[2]-0.454*w.t[1]
  for (i in 3:10000) {
    serie.sim.en[i]=3.05* 10^{-4}+ 0.445*(serie.sim.en[i-1]-3.05* 10^{-4}) -
      0.933*(serie.sim.en[i-2]-3.05* 10^{-4})+ w.t[i]-0.454*w.t[i-1]+0.95 * w.t[i-2]
  }
  return(serie.sim.en)
}


# Simulated returns per stock
serie.sim.en=funcSim.en(sim[,6])  # Simulated series of EN
```

- Teleperformance (TEP)

```
# Function to compute the simulated Schneider returns
funcSim.tep<-function(copSim){
  serie.sim.tep<-rep(0, 10000)
  w.t<- copSim * sim.sigma.tep
  serie.sim.tep[1]=1.13* 10^{-3}+w.t[1]
  serie.sim.tep[2]=1.13* 10^{-3}+ 0.053*(serie.sim.tep[1]-1.13* 10^{-3})+ w.t[2]-0.116*w.t[1]
  for (i in 3:10000) {
    serie.sim.tep[i]=1.13* 10^{-3}+ 0.053*(serie.sim.tep[i-1]-1.13* 10^{-3})+
      0.677*(serie.sim.tep[i-2]-1.13* 10^{-3})+ w.t[i]-0.116*w.t[i-1]-0.7 * w.t[i-2]
  }
  return(serie.sim.tep)
}


# Simulated returns per stock
serie.sim.tep=funcSim.tep(sim[,7])  # Simulated series of TEP
```

- Engie

```
# Function to compute the simulated Schneider returns
funcSim.engie<-function(copSim){
  serie.sim.engie<-rep(0, 10000)
  w.t<- copSim * sim.sigma.engie
  serie.sim.engie[1]=1.448* 10^{-4}+w.t[1]
  serie.sim.engie[2]=1.448* 10^{-4}+ 0.73*(serie.sim.engie[1]-1.448* 10^{-4})+ w.t[2]
  for (i in 3:10000) {
    serie.sim.engie[i]=1.448* 10^{-4}+ 0.73*(serie.sim.engie[i-1]-1.448* 10^{-4})+ w.t[i]
  }
  return(serie.sim.engie)
}


# Simulated returns per stock
serie.sim.engie=funcSim.engie(sim[,8])  # Simulated series of TEP
```

- Michelin (ML)

```
# Function to compute the simulated Schneider returns
funcSim.ml<-function(copSim){
  serie.sim.ml<-rep(0, 10000)
  w.t<- copSim * sim.sigma.ml
  serie.sim.ml[1]=2.48* 10^{-4}+w.t[1]
  for (i in 2:10000) {
    serie.sim.ml[i]=2.48* 10^{-4}+ w.t[i]-0.02*w.t[i-1]
  }
  return(serie.sim.ml)
}


# Simulated returns per stock
serie.sim.ml=funcSim.ml(sim[,9])  # Simulated series of ML
```

- Société Générale (GLE)

```
# Function to compute the simulated Schneider returns
funcSim.gle<-function(copSim){
  serie.sim.gle<-rep(0, 10000)
  w.t<- copSim * sim.sigma.gle
  serie.sim.gle[1]=-3.18* 10^{-4}+w.t[1]
  serie.sim.gle[2]=-3.18* 10^{-4}+ 0.61*(serie.sim.gle[1]+3.18* 10^{-4})+ w.t[2]-0.52*w.t[1]
  for (i in 3:10000) {
    serie.sim.gle[i]=-3.18* 10^{-4}+ 0.61*(serie.sim.gle[i-1]+3.18* 10^{-4})-
      0.35*(serie.sim.gle[i-2]+3.18* 10^{-4})+ w.t[i]-0.52*w.t[i-1]+0.3 * w.t[i-2]
  }
  return(serie.sim.gle)
}


# Simulated returns per stock
serie.sim.gle=funcSim.gle(sim[,10])  # Simulated series of GLE
```

Portfolio VaR

VaR by methods

```
alpha<-0.05

portfolio.model=cbind(serie.sim.gle, serie.sim.ml, serie.sim.engie, serie.sim.tep,
                      serie.sim.en, serie.sim.bnp, serie.sim.vie, serie.sim.ca,
                      serie.sim.K, serie.sim.S)

# Portfolio simulated return
ret_sim.tda<-as.numeric(as.matrix(portfolio.model) %*% weight.tda) # Computing the portfolio return on s
ret_sim.mc<-as.numeric(as.matrix(portfolio.model) %*%weight.mc) # Computing the portfolio return on sim


# Risk measurement
VaR.bin1.tda<-quantile(ret_sim.tda, alpha) # VaR tda
VaR.bin1.mc<-quantile(ret_sim.mc, alpha) # VaR mc


ES.bin1.tda<-mean(ret_sim.tda[ret_sim.tda<VaR.bin1.tda])  # ES
ES.bin1.mc<-mean(ret_sim.mc[ret_sim.mc<VaR.bin1.mc])  # ES

# VaR data frame
Method.var<-data_frame(Methods=c('TDA', 'MC'), VaR=c(VaR.bin1.tda, VaR.bin1.mc),
                       ES=c(ES.bin1.tda, ES.bin1.mc))

# VaR output
round(VaR.bin1.tda, 6)
```

```
##       5%
## 0.000625
```

```
round(VaR.bin1.mc, 6)
```

```
##      5%
```

```
## 0.00051
```

- Save the data in external csv file

```r
write.table(x=Method.var, file="Method.var.csv" , sep=";", dec = ",") # Save VaR for methods
```