Managing risk on the CaC 40 stock index

Miradain Atontsa (miradain.atontsan@gmail.com)

10/7/2020

Introduction

This risk analysis report is part of a series of studies on the CaC40 stock market index. In the first report, topological data analysis theory was used as an Enhanced Indexing approach to build portfolios with assets taken from the CaC40 index, so that each portfolio has a better profit than the index. reference. Namely, we have built two portfolios. The first named "Bin1" contains: Société Générale, Michelin, Engie, Teleperformance, Bouygues, BNP Paribas, Veolia Environ, Carrefour, Kering and finally Schneider electic. The second asset named "Bin3" includes: Pernod Ricard, Atos, Orange, Airbus, Accor, Peugeot, Sodexo, Saint-Gobain, Crédit Agricole and Thales.

The aim of this analysis is to model the return of each portfolio and then to compare them in terms of the quantitative risk generated. We will see that bin1 is suitable for people with a high risk profile while bin3 is suitable for people with a low risk profile.

Methodology

- Analysis of risk for Bin1 In the analysis of the Bin1 portfolio, our optimization process (in the first report) showed that the most important assets were the assets of Kering (95.4%) and Schneider (3.56%). We will therefore only consider a portfolio with these assets. In other words, we will consider a portfolio with Kering (0.96) and Schneider (0.04%). We will carry out univariate ARMA-GARCH (1,1) models for each asset and perform a Monte Carlo simulation to obtain the conditional volatilities. In addition, we will use the copula approach to estimate and simulate the model residuals. This will permit to update the simulated values of the ARMA-GARCH model and then to calculate a non-parametric Value-at-Risk (VaR) and expected shortfall (ES) of the portfolio.
- Analysis of risk for Bin3 In the analysis of Bin3 portfolio, our optimization process (in the first report) showed that the most important asset was the Pernod Ricard (99.1%). We will therefore only consider a portfolio with this single asset. In other words, we consider in this risk analysis a portfolio with Pernod Ricard (100%). Thus our risk analysis on this portfolio will follow an univariate ARMA-GARCH(1,1) model. At the end of the process, we will perform a Monte Carlo simulation to compute a non-parametric Value-at-Risk (VaR) and Expected Shortfall (ES).

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('KER.PA', 'SU.PA', 'RI.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]
```

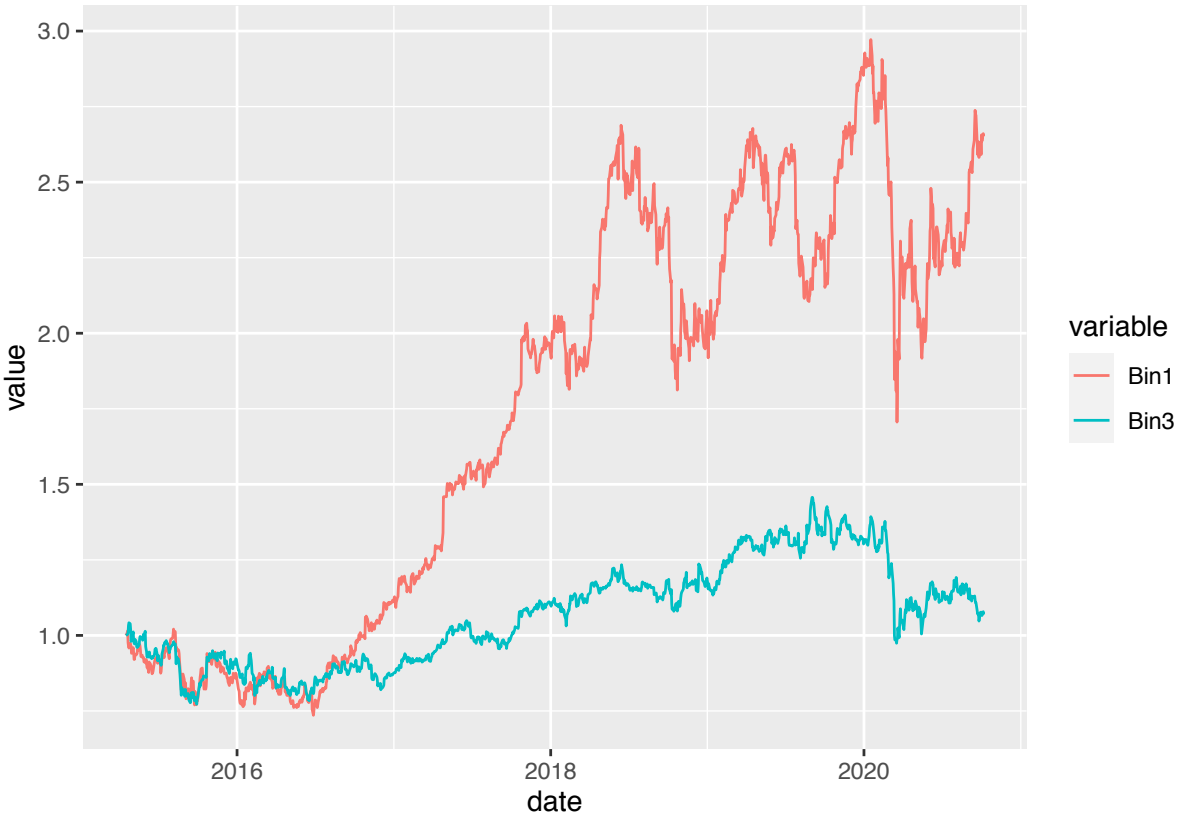
Data processing

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

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

# Extracting close price data
data <- data_frame(date=df_Kering$ref.date,
                   Kering=df_Kering$price.close,
                   Schneider= df_Schneider$price.close,
                   Ricard=df_Ricard$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$Ricard<-append( returns_qrmtools(data$Ricard ,method='logarithmic'), 0, after = 0)

# Compute the cumulative return
ret_Bin1 <- as.numeric(cbind(data$Kering, data$Schneider) %*% c(0.96, 0.04))
ret_Bin3<-as.numeric(data$Ricard)

cret_Bin1<- cumprod(1+ret_Bin1)
cret_Bin3<-cumprod(1+ret_Bin3)
```

Plot portfolio cummulative returns

We now plot the cummulative returns of portfolios bin1 and bin3.

```
data_melted<-reshape2::melt(data_frame(date=data$date, Bin1=cret_Bin1, Bin3=cret_Bin3), id.var='date')
ggplot(data_melted, aes(x=date, y=value, col=variable))+geom_line()
```



The plot shows that bin1 tends to be more unstable that bin3. In the rest of this study, we will build the risk model of these portfolios. While we will conduct a multivariate GARCH model for bin1, a univariate GARCH model for bin3 will be sufficient.

Fitting univariate GARCH (using rugarch)

A GARCH approach consists of modelling the volatility of a stock price. A GARCH(1,1) model is given by

$$r_t = \mu + \varepsilon_t$$

where ε_t needs to be developed for each asset. We have to choose the mean model and the variance model. Let's choose the most appropriate mean model first, using the "autoarfima" function to test all combinations. Note that this estimation will take a long time if done directly and as such we can use of the parallel library which allows our computer to do similar parallel computations at the same time so as to save time.

Case of Kering

- Mean model selection

```r
# 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 -5.012577         1
## 2  2  0    1      0 -5.012437         1
## 3  1  0    1      0 -5.012316         1
## 4  0  1    1      0 -5.012170         1
## 5  0  2    0      0 -5.011895         1
## 6  0  0    1      0 -5.011789         1
```

These analysis suggest a ARMA(0,2) model for Kering.

- Variance model selection

We loop for best fitting GARCH model

```r
# Kering case
models=1:9
model.list=list()

for (p in models) {
  garch<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "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")[p]
  )

  garchfit1<-ugarchfit(spec=garch, data=as.numeric(data$Kering))

  model.list[[p]]=garchfit1

}

names(model.list)<-c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")
fit.mat<- sapply(model.list, infocriteria)

rownames(fit.mat)=rownames(infocriteria(model.list[[1]]))
fit.mat
```

```
##                   norm     snorm       std      sstd       ged      sged
## Akaike       -5.126517 -5.125943 -5.284483 -5.283734 -5.268144 -5.267711
## Bayes        -5.104055 -5.099737 -5.258277 -5.253784 -5.241938 -5.237761
## Shibata      -5.126554 -5.125993 -5.284533 -5.283798 -5.268193 -5.267776
## Hannan-Quinn -5.118121 -5.116147 -5.274687 -5.272538 -5.258348 -5.256516
##                    nig      ghyp       jsu
## Akaike       -5.280010 -5.282346 -5.282765
## Bayes        -5.250060 -5.248653 -5.252815
## Shibata      -5.280074 -5.282428 -5.282830
## Hannan-Quinn -5.268814 -5.269751 -5.271570
```

The table suggests that the std,sstd, ged, ghyp and jsu models perform the best (lowest AIC). We will consider for the sequel the student t distribution. In conclusion, we will use the following as the GARCH

model of the log Kering return

$$r_t = \mu + w_t + \phi_1 w_{t-1} + \phi_2 w_{t-2}$$

and

$$w_t = \sigma_t . z_t$$

where the w_t are white noise and $z_t \sim t(d)$ with an unknown degree of freedom d.

- 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
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      1.276221e-03 3.573392e-04   3.571454 3.550048e-04
## ma1    -4.522652e-02 2.508250e-02  -1.803111 7.137084e-02
## ma2    -3.352344e-02 2.601589e-02  -1.288576 1.975456e-01
## omega   7.478547e-06 4.875913e-06   1.533774 1.250854e-01
## alpha1  6.000478e-02 7.275394e-03   8.247634 2.220446e-16
## beta1   9.257545e-01 8.899724e-03 104.020593 0.000000e+00
## shape   3.775192e+00 2.697270e-01  13.996342 0.000000e+00
```

This table says that our fitted model for the Kering data set is

$$r_t = 1.3 \times 10^{-3} + w_t - 0.05 w_{t-1} - 0.03 w_{t-2}$$

where

$$w_t = \sigma_t . z_t$$

$$\sigma_t^2 = 7.5 \times 10^{-6} + 0.06 \varepsilon_{t-1}^2 + 0.93 \sigma_{t-1}^2$$

and

$$z_t \sim t(3.78).$$

- Non-parametric 95% VaR and ES for Kering

Now we are going to compute the Value at Risk (VaR) and the Expected Shortfall (ES), using a non-parametric method. More precisely, we will simulate values from the previously build ARMA(0,2)-GARCH(1,1) model (with t distributed innovations): garchfit.K.

```
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
```

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

```
##        5%
## -0.030931
```

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

```
## [1] -0.058284
```

In other words, the non-parametric VaR for the Kering stock is: -0.030 while its ES is -0.058.

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  1  2    0      0 -5.288706         1
## 2  2  1    0      0 -5.288703         1
## 3  0  0    1      0 -5.288692         1
## 4  0  1    0      0 -5.288383         1
## 5  1  0    0      0 -5.288341         1
## 6  1  2    1      0 -5.287913         1
```

These analysis suggest a ARMA(1,2) model for Schneider.

- Variance model selection

We loop for best fitting GARCH model.

```r
# Schneider case
models=1:9
model.list=list()

for (p in models) {
  garch<-ugarchspec(
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "apARCH")[1], garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(1,2), include.mean=TRUE),
  distribution.model = c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")[p]
  )

  garchfit1<-ugarchfit(spec=garch, data=as.numeric(data$Schneider))

  model.list[[p]]=garchfit1

}

names(model.list)<-c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")
fit.mat<- sapply(model.list, infocriteria)
```

```
rownames(fit.mat)=rownames(infocriteria(model.list[[1]]))
fit.mat
```

```
##                  norm     snorm       std      sstd       ged      sged
## Akaike       -5.494437 -5.503830 -5.583109 -5.587439 -5.576401 -5.581751
## Bayes        -5.468231 -5.473880 -5.553159 -5.553746 -5.546451 -5.548058
## Shibata      -5.494487 -5.503895 -5.583174 -5.587521 -5.576466 -5.581833
## Hannan-Quinn -5.484641 -5.492634 -5.571913 -5.574844 -5.565206 -5.569157
##                   nig      ghyp       jsu
## Akaike       -5.588254 -5.586957 -5.588476
## Bayes        -5.554560 -5.549520 -5.554782
## Shibata      -5.588335 -5.587058 -5.588558
## Hannan-Quinn -5.575659 -5.572963 -5.575881
```

The table suggests that the std,sstd, ged, sged,nig, ghyp and jsu models perform the best (lowest AIC). We will consider for the sequel the student t distribution. In conclusion, we will use the following as the GARCH model of the log Schneider return

$$r_t = \mu + \alpha_1.(r_{t-1} - \mu) + w_t + \phi_1 w_{t-1} + \phi_2 w_{t-2}$$

and

$$w_t = \sigma_t.z_t$$

where the w_t are white noise and $z_t \sim t(d)$ with an unknown degree of freedom d.

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

```
# 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],
                        garchOrder=c(1,1)),
  mean.model = list( armaOrder=c(1,2), 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      8.568322e-04 2.807414e-04  3.0520338 2.272965e-03
## ar1     7.286265e-01 1.822814e-01  3.9972628 6.407915e-05
## ma1    -7.384753e-01 1.838709e-01 -4.0162715 5.912614e-05
## ma2    -3.525949e-02 2.977466e-02 -1.1842116 2.363293e-01
## omega   6.143067e-06 1.170591e-05  0.5247833 5.997338e-01
## alpha1  7.092584e-02 3.219738e-02  2.2028450 2.760567e-02
## beta1   9.102825e-01 9.897868e-03 91.9675333 0.000000e+00
## shape   4.596475e+00 2.117752e-01 21.7044967 0.000000e+00
```

This table says that our fitted model for the Schneider data set is

$$r_t = 8.57 \times 10^{-4} + 0.73(r_{t-1} - 8.57 \times 10^{-4}) + w_t - 0.74 w_{t-1} - 0.04 w_{t-2}$$

where $w_t = \sigma_t.z_t$,

$$\sigma_t^2 = 6.14 \times 10^{-6} + 0.07\varepsilon_{t-1}^2 + 0.91\sigma_{t-1}^2$$

$$z_t \sim t(4.60).$$

- Non-parametric 95% VaR and ES for Schneider

Now we are going to compute the Value at Risk (VaR) and the Expected Shortfall (ES), using a non-parametric method. More precisely, we will simulate values from the previously build ARMA(1,2)-GARCH(1,1) model (with t distributed innovations): garchfit.S.

```
# 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%
## -0.024142
```

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

```
## [1] -0.037816
```

In other words, the non-parametric VaR for the Schneider stock is: -0.024 while its ES is -0.037.

Case of Ricard

- Mean model selection

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

```
##   AR MA Mean ARFIMA       AIC converged
## 1  0  1    0      0 -5.866247         1
## 2  1  0    0      0 -5.866247         1
## 3  0  0    1      0 -5.866231         1
## 4  2  1    0      0 -5.865222         1
## 5  0  1    1      0 -5.864921         1
## 6  1  0    1      0 -5.864921         1
```

These analysis suggest a ARMA(0,1) model for Ricard.

- Variance model selection

We loop for best fitting GARCH model.

```
# Schneider case
models=1:9
model.list=list()

for (p in models) {
  garch<-ugarchspec(
```

```
  variance.model = list(model=c("sGARCH", "gjrGARCH", "eGARCH", "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")[p]
  )

  garchfit1<-ugarchfit(spec=garch, data=as.numeric(data$Ricard))

  model.list[[p]]=garchfit1

}

names(model.list)<-c("norm", "snorm", "std", "sstd", "ged", "sged", "nig", "ghyp", "jsu")
fit.mat<- sapply(model.list, infocriteria)

rownames(fit.mat)=rownames(infocriteria(model.list[[1]]))
#xtable(fit.mat)
fit.mat
```

```
##                   norm     snorm       std      sstd       ged      sged
## Akaike       -6.012441 -6.015369 -6.095748 -6.094326 -6.086251 -6.084833
## Bayes        -5.993722 -5.992907 -6.073285 -6.068120 -6.063788 -6.058627
## Shibata      -6.012466 -6.015406 -6.095784 -6.094376 -6.086287 -6.084883
## Hannan-Quinn -6.005444 -6.006973 -6.087351 -6.084530 -6.077854 -6.075037
##                    nig      ghyp       jsu
## Akaike       -6.093978 -6.093985 -6.094634
## Bayes        -6.067772 -6.064036 -6.068428
## Shibata      -6.094028 -6.094050 -6.094683
## Hannan-Quinn -6.084182 -6.082790 -6.084838
```

The table suggests that the std,sstd, ged, ghyp and jsu models perform the best (lowest AIC). We will consider for the sequel the student t distribution. In conclusion, we will use the following as the GARCH model of the log Ricard return

$$r_t = \mu + w_t + \phi_1 w_{t-1}$$

and

$$w_t = \sigma_t . z_t$$

where the w_t are white noise and $z_t \sim t(d)$ with an unknown degree of freedom d.

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

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

garch.R<-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.R=ugarchfit(spec = garch.R, data=as.numeric(data$Ricard))

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

```
##            Estimate   Std. Error    t value     Pr(>|t|)
```

```
## mu     4.899033e-04 2.656850e-04  1.8439253 6.519403e-02
## ma1    5.341021e-03 2.669699e-02  0.2000608 8.414330e-01
## omega  1.898758e-06 1.246379e-06  1.5234190 1.276539e-01
## alpha1 4.824938e-02 1.060867e-02  4.5481095 5.412998e-06
## beta1  9.413899e-01 1.187451e-02 79.2782026 0.000000e+00
## shape  4.858715e+00 5.526427e-01  8.7917836 0.000000e+00
```

This table says that our fitted model for the Ricard data set is

$$r_t = 5 \times 10^{-4} + w_t - 0.01 w_{t-1}$$

where

$$w_t = \sigma_t . z_t$$

with

$$\sigma_t^2 = 2 \times 10^{-6} + 0.05 \varepsilon_{t-1}^2 + 0.94 \sigma_{t-1}^2$$

and

$$z_t \sim t(4.86).$$

- Non-parametric 95% VaR and ES for Ricard

Now we are going to compute the Value at Risk (VaR) and the Expected Shortfall (ES), using a non-parametric method. More precisely, we will simulate values from the previously build ARMA(0,2)-GARCH(1,1) model (with t distributed innovations): garchfit.R.

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


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

```
##        5%
## -0.021451
```

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

```
## [1] -0.032502
```

In other words, the non-parametric VaR for the Ricard stock is: -0.021 while its ES is -0.032.

Fitting multivariate GARCH for the couple (Kering, Schneider)

We will further need to compare the risk measures of bin1 and bin3. While we already have the values for bin3 through the previous computations, we are not done yet for bin1. We know in fact that its ES will be less that $ ES.K+ES.S= -0.094$ but without being more precised than that. The goal of this multivariate analysis is to compute the risk measures of the portfolio.

Test of normality

We use the Jarque Bera test to evaluate if our financial data is normaly distributed. This is a basic step in risk management as this will permit to avoid certain behavior such as relying on correlations to evaluate VaR.

This test rely on the following statistic:

$$BJ = n(\frac{S^2}{6} + \frac{(K-3)^2}{24})$$

where S and K are respectively model estimators of the Skewness and Kurtosis.

```
#The Jarque-Bera test of normality
JB_Kering<-jarque.bera.test(data$Kering)
JB_Schneider<-jarque.bera.test(data$Schneider)

JB_Kering
```

```
##
##  Jarque Bera Test
##
## data:  data$Kering
## X-squared = 1589.3, df = 2, p-value < 2.2e-16
```

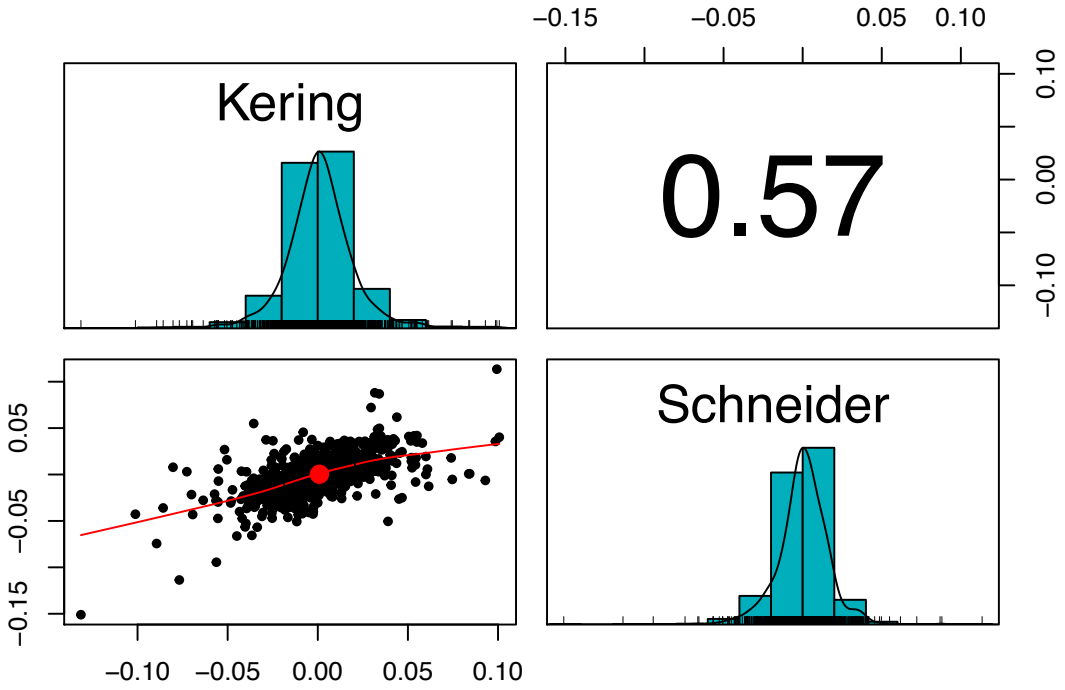
```
JB_Schneider
```

```
##
##  Jarque Bera Test
##
## data:  data$Schneider
## X-squared = 4844.4, df = 2, p-value < 2.2e-16
```

The null hypothesis is rejected in the two cases as the p-value is extremly small. We are then unable to claim that our data from Kering and Schneider are normaly distributed. This is not surprising as financial data are rarely Gaussian.

Plotting the distributions

```
pairs.panels(data[c("Kering", "Schneider")],
             method = "spearman", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```



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))}
```

Model residual

In the previous step, we have built ARMA-GARCH models with the residual

$$w_t = \sigma_t z_t$$

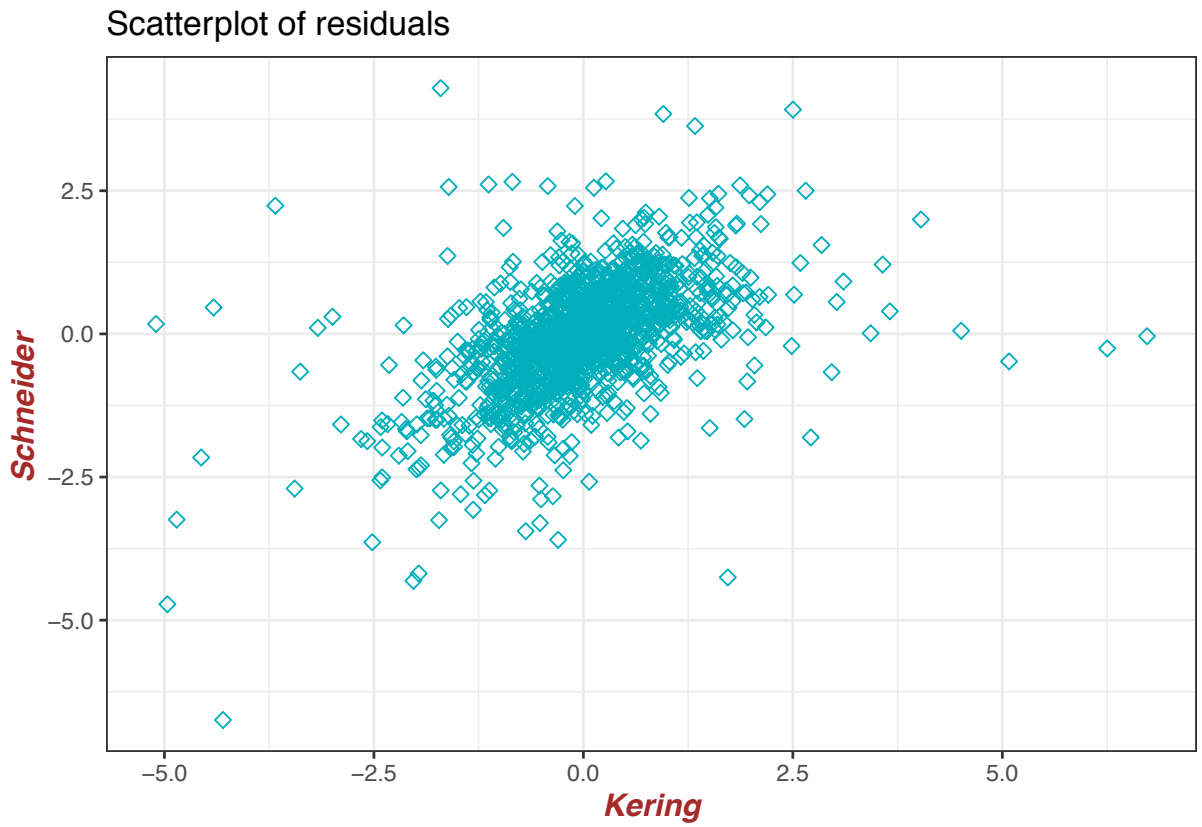
where $z_t \sim t(d)$.

```
wt.K<-residuals(garchfit.K)    # Ordinary resids
sigma.K<-sigma(garchfit.K)     # Conditional resids
zt.K<- residuals( garchfit.K, standardize=TRUE)  # Standardized resids
# all.equal(wt.K/sigmat.K, zt.K) # Checking: wt=sigmat.zt

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

- Scatterplot of residuals

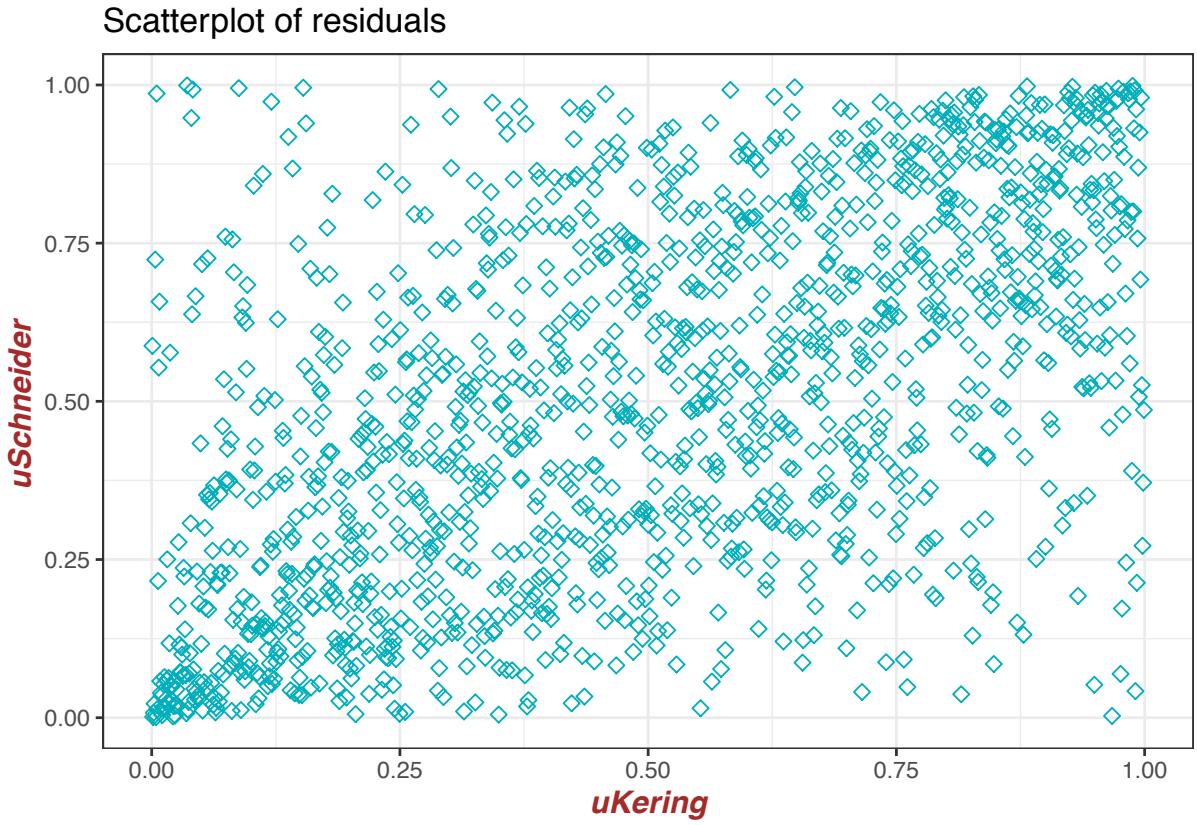
```
theme_set(
  theme_bw() +
    theme(legend.position = "top")
  )
```

```
Res_data<-data_frame(Kering_res=zt.K, Schneider_res=zt.S)
# Initiate a ggplot
b <- ggplot(Res_data[c("Kering_res", "Schneider_res")], aes(x = Kering_res, y = Schneider_res))



# Basic scatter plot
b + geom_point(color = "#00AFBB", size = 2, shape = 23)+
  theme(axis.title=element_text(face="bold.italic", size="12", color="brown"), legend.position="top") +
  labs(title="Scatterplot of residuals", x="Kering", y="Schneider ")
```



- Scatterplot of residuals ranks (u_{1t}, u_{2t})

```
theme_set(
  theme_bw() +
    theme(legend.position = "top")
  )

uRes_data<-data_frame(uKering=pobs(zt.K), uSchneider=pobs(zt.S))
# Initiate a ggplot
b <- ggplot(uRes_data[c("uKering", "uSchneider")], aes(x = uKering, y = uSchneider))

# Basic scatter plot
b + geom_point(color = "#00AFBB", size = 2, shape = 23) +
  theme(axis.title=element_text(face="bold.italic", size="12", color="brown"), legend.position="top") +
  labs(title="Scatterplot of residuals", x="uKering", y="uSchneider ")
```



Copulas

Fit the Gaussian copula

```
library(copula)
# Correlation between marginal distribution associated to Hering and Schneider data
correlation=cor(uRes_data$uKering,uRes_data$uSchneider, method = 'spearman')

set.seed(100)

# Creating the normal copula object
normal_Cop <- normalCopula(param=c(correlation), dim = 2, dispstr = "un")

# fit Gaussian copula: estimating rho along the way
fit.gaussian<-fitCopula(normal_Cop, uRes_data, method="ml",
                        start=c(correlation), optim.control=list(maxit=2000) )


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

Fit a t-copula

```
dist_param<-apply (data[c("Kering", "Schneider")], 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]))

# Creating the t-copula object
tcop<- tCopula(param = c(correlation), dim=2, 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)
```

Fit a Clayton copula

```
# Define the copula object
ccop<-claytonCopula(dim=2)

# Fitting the t-copula:estimating theta along the way
fit.c<-fitCopula(ccop, uRes_data, method="ml",
                 optim.control=list(maxit=2000))

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

Fit a Gumbel copula

```
# Define the copula object
gcop<-gumbelCopula(dim=2)

# Fitting the t-copula:estimating theta along the way
fit.g<-fitCopula(gcop, uRes_data, method=c("ml"),
                 optim.control=list(maxit=2000))

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

Fit a Frank copula

```
# Define the copula object
fcop<-frankCopula(dim=2)

# Fitting the t-copula:estimating theta along the way
fit.f<-fitCopula(fcop, uRes_data, method="ml",
                 optim.control=list(maxit=2000))

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

- Recap: Table for copulas

```
Copulas<-data_frame(copula=c( 'Gaussian', 'Student', 'Clayton', 'Gumbel', 'Frank'),
                    AIC=c(fit.gaussian.aic,fit.t.aic,fit.c.aic, fit.g.aic, fit.f.aic  )
```

```
                    )
Copulas
```

```
## # A tibble: 5 x 2
##   copula     AIC
##   <chr>    <dbl>
## 1 Gaussian -496.
## 2 Student  -594.
## 3 Clayton  -507.
## 4 Gumbel   -471.
## 5 Frank    -537.
```

The best copulas is the 'Student' copula since it has a lowest value.

Monte Carlo simulation

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

```
best_copula<-tCopula(param = c(fit.t@estimate[1]),df=fit.t@estimate[2], dim=2, dispstr = "un")


copula_distribution<- mvdc(copula = best_copula, margins = c("ct","ct"),
  paramMargins = list(
        list(m=dist_param$Kering$estimate[1], s=dist_param$Kering$estimate[2], df=dist_param$Kering$esti
        list(m=dist_param$Schneider$estimate[1], s=dist_param$Schneider$estimate[2], df=dist_param$Schn
                            ) )
```

Simulation

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

# 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]=w.t[1]
  serie.sim.K[2]=w.t[2]-0.04*w.t[1]
  for (i in 3:10000) {
    serie.sim.K[i]=w.t[i]-0.04*w.t[i-1]-0.03 * w.t[i-2]
  }
  return(serie.sim.K)
}

# Function to compute the simulated Schneider returns
funcSim.S<-function(copSim){
  serie.sim.S<-rep(0, 10000)
  w.t<- copSim * sim.sigma.K
```

```
  serie.sim.S[1]=8.57* 10^{-4}+w.t[1]
  serie.sim.S[2]=8.57* 10^{-4}+ 0.73*(serie.sim.S[1]-8.57* 10^{-4})+ w.t[2]-0.74*w.t[1]
  for (i in 3:10000) {
    serie.sim.S[i]=8.57* 10^{-4}+ 0.73*serie.sim.S[i-1]+ w.t[i]-0.74*w.t[i-1]-0.04 * w.t[i-2]
  }
  return(serie.sim.S)
}

# Simulated returns per stock
serie.sim.K=funcSim.K(sim[,1])   # Simulated series of Kering
serie.sim.S=funcSim.S(sim[,2])  # Simulated series of Schneider

# Portfolio simulated return
ret_sim <- as.numeric(cbind(serie.sim.K, serie.sim.S) %*% c(0.96, 0.04)) # Computing the portfolio retu

# Risk measurement
VaR.bin1<-quantile(ret_sim, alpha) # VaR
ES.bin1<-mean(ret_sim[ret_sim<VaR.bin1])  # ES
round(VaR.bin1, 6)
```

```
##        5%
## -0.000497
```

```
round(ES.bin1, 6)
```

```
## [1] -0.000989
```

In other words, the non-parametric VaR for the bin1 portfolio is: -0.0005 while its ES is -0.001.

Conclusion

In this study, we compared the risk of investing in portfolios made out of the CaC 40 index. The first bin1 portfolio had two assets: Kering and Schneider. Its nonparametric VaR was -0.0005 and its expected deficit was -0.001. On the other hand, the second bin3 portfolio which contained only the Pernod Ricard asset had a VaR = -0.021 while ES = -0.032. Based on this analysis, we can conclude that investing in Bin1 is riskier than investing in bin3.

Declaration of Competing Interest

The author declares that he has no known competing financial interest or personal relationships that could have appeared to influence the work reported in this tutorial.