Fitting copulas

The goal of this project is to study a concrete example of multivariate stock returns using copula estimation. More specifically, we will:

- Fit univariate models for each margin individually.
- Fit a copula to account for the dependence between the margins.
- Simulate new data following the fitted model and use it to make predictions about the aggregated loss.

0) Setup

The first step is to load the necessary libraries.

```
library(rugarch)
library(xts)
library(ADGofTest)
library(qqtest)
library(copula)
library(qrmdata)
library(qrmtools)

set.seed(123) # for reproducibility
```

1) Data Preparation:

The data consists of an `xts` object containing adjusted close prices of the constituents of the S&P 500 index. The data hs been obtained from Yahoo Finance as of 2016-01-03, via the function `get_data()` from the package **qrmtools**. The constituents data ranges from the first date at least one of the constituents is available (with missing data if not available) to 2015-12-31.

We will only consider 5 constituents of the index: Intel, Qualcomm, Google, Apple and Microsoft from January $3^{rd}, 2007$ to December $31^{st}, 2009$.

First, we create a matrix `S` containing the observations of interest and check that it doesn't contain missing data.

```
# load the constituents data of the S&P 500
data("SP500_const")
stocks <- c("INTC", "QCOM", "GOOGL", "AAPL", "MSFT")
time <- c("2007-01-03", "2009-12-31")

#observations of interest:
S <- SP500_const[paste0(time, collapse = "/"), stocks]

## Check for missing data
any(is.na(S))
```

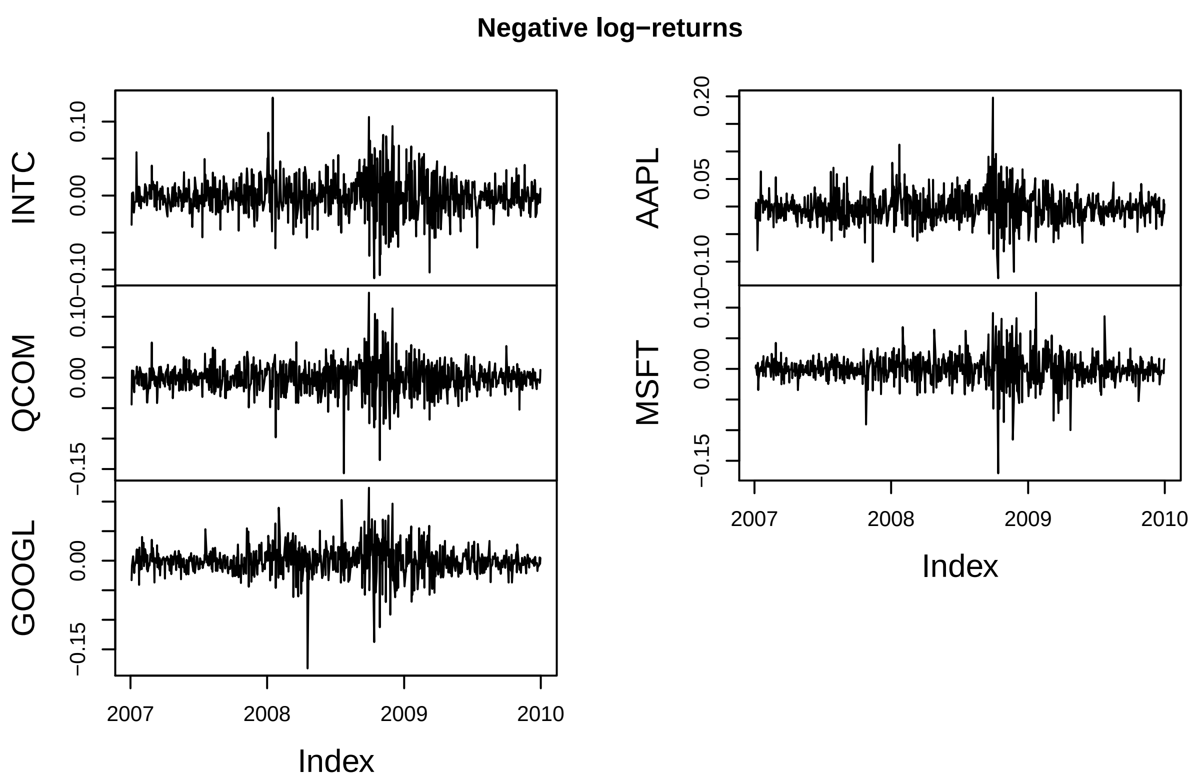
```
## [1] FALSE
```

Then, we compute the negative log-returns (representing financial losses)

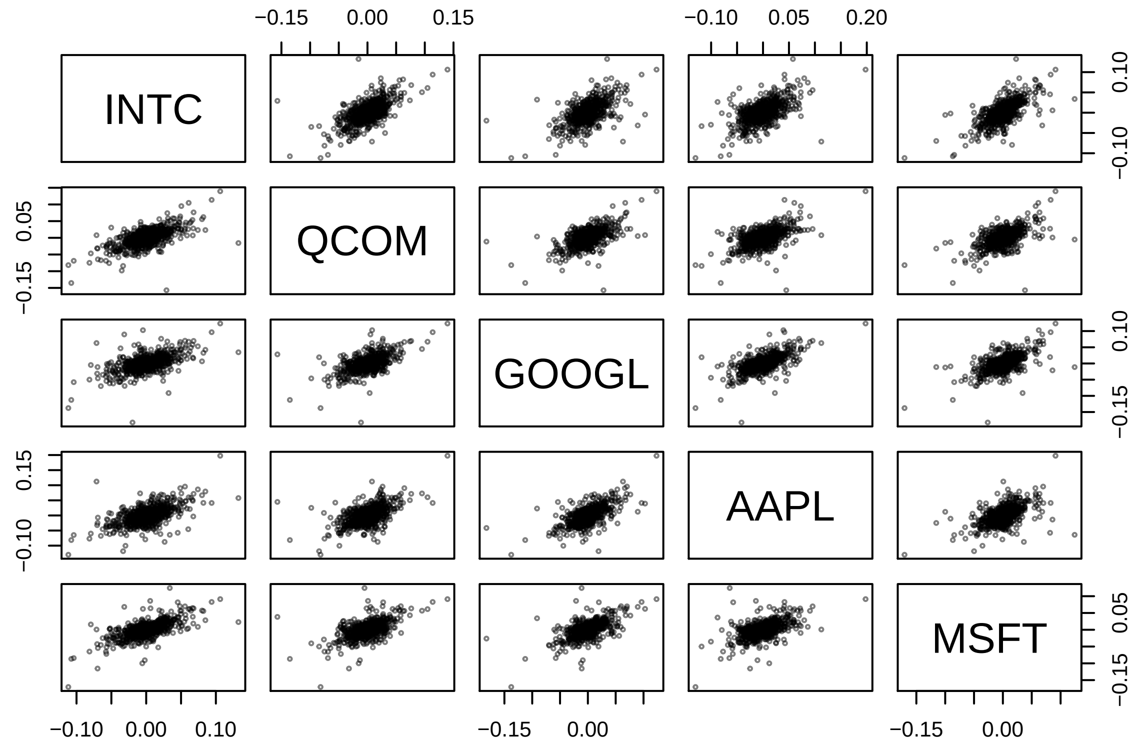
```r
## Build -log-returns
X <- -returns(S,method = 'logarithm')

# sample size
n <- nrow(X)

# dimension
p <- ncol(X)
```

We now create some exploratory plots to study their evolution in time (acf of X and X^2), their marginal distributions, their joint behavior.

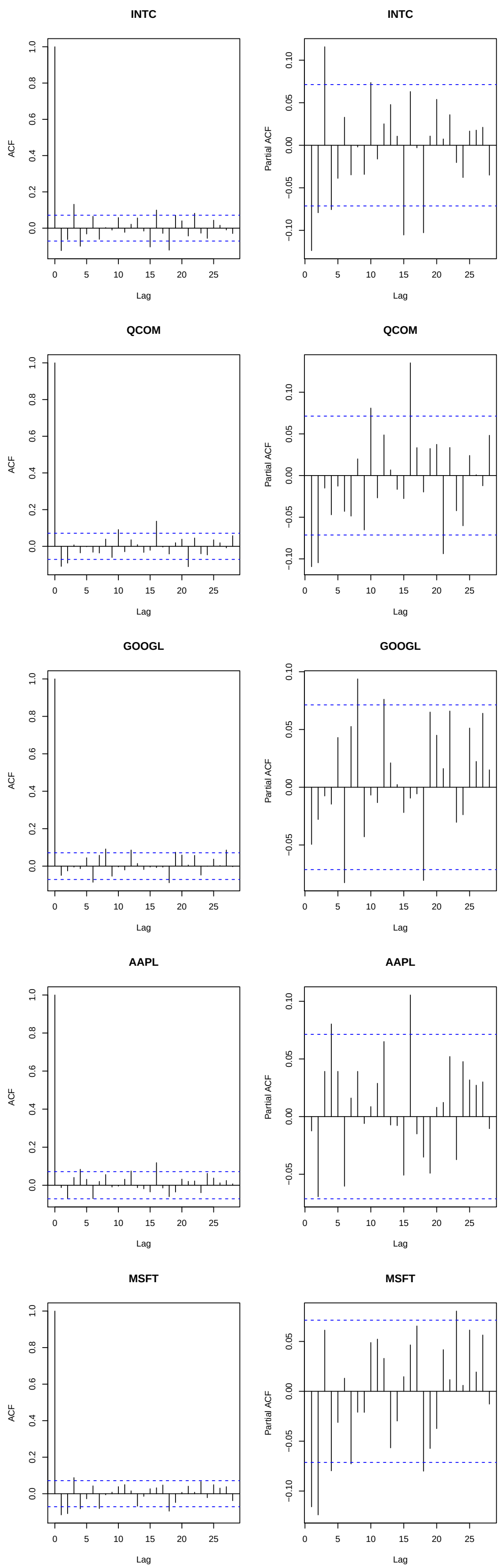
```r
## Basic plot: use functions plot.zoo, pairs, histograms
# eventually draw acf, pacf for each margin

plot.zoo(X, main = "Negative log-returns")
```



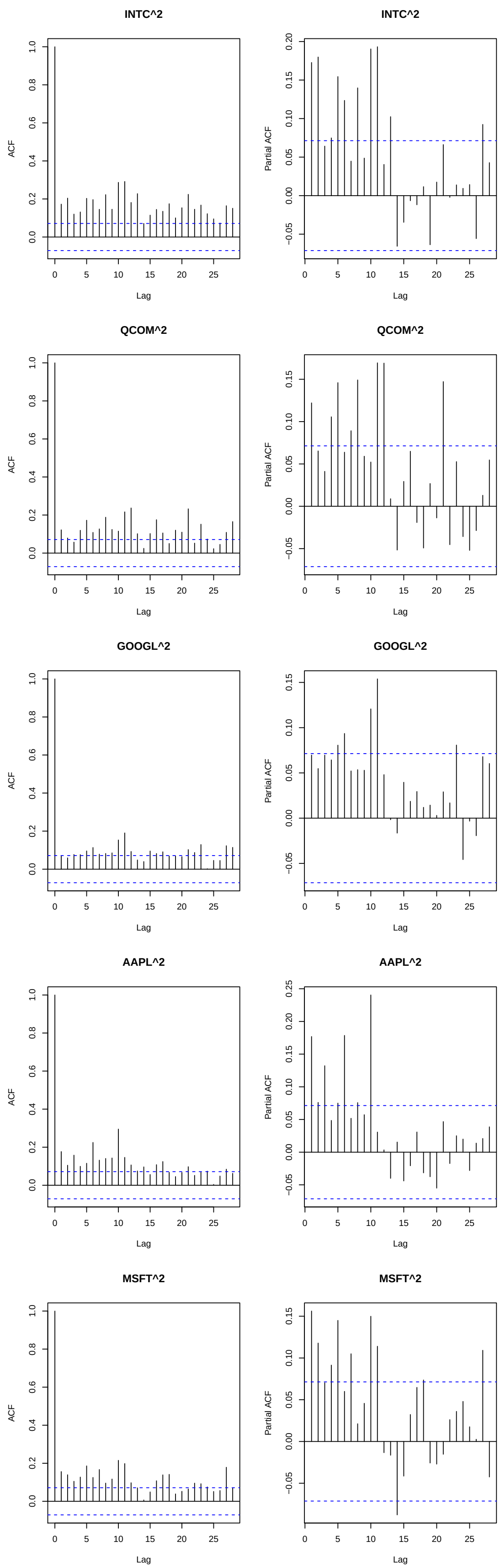

```r
pairs(as.matrix(X),cex = 0.4, col = adjustcolor("black", alpha.f = 0.5))
```




```
par(mfrow=c(5,2))
for (k in 1:5)
{
  acf(X[,k],main=colnames(X)[k])
  pacf(X[,k],main=colnames(X)[k])
}
```


INTC
ACF
Lag
INTC
Partial ACF
Lag
QCOM
ACF
Lag
QCOM
Partial ACF
Lag
GOOGL
ACF
Lag
GOOGL
Partial ACF
Lag
AAPL
ACF
Lag
AAPL
Partial ACF
Lag
MSFT
ACF
Lag
MSFT
Partial ACF
Lag

```
par(mfrow=c(5,2))
for (k in 1:5)
{
  acf(X[,k]^2,main=paste0(colnames(X)[k],'^2'))
  pacf(X[,k]^2,main=paste0(colnames(X)[k],'^2'))
}
```


INTC^2
ACF
Partial ACF
Lag
INTC^2
QCOM^2
ACF
Partial ACF
Lag
QCOM^2
GOOGL^2
ACF
Partial ACF
Lag
GOOGL^2
AAPL^2
ACF
Partial ACF
Lag
AAPL^2
MSFT^2
ACF
Partial ACF
Lag
MSFT^2

We want to model the dependence in the mean and the variance of the process $(X_t)_{t>0}$.

The autocorrelation function of $(X_t)_{t>0}$ shows some evidence of serial correlation at small lags. This suggest that an ARMA(1,1) could be suitable for modelling.

The autocorrelation functions of the squared components $(X_t^2)_{t>0}$, representing the spot volatility (under the assumption of zero mean) also shows us that the volatility is heteroskedastic. We can take this into account through a GARCH(1,1) process.

In fact, such a model is often used for time series of financial returns.

2) Fitting marginal models

We consider an ARMA(1,1)-GARCH(1,1) with standardized t residuals. This model is fitted by using the function `fit_ARMA_GARCH` in the package **qrmtools**.

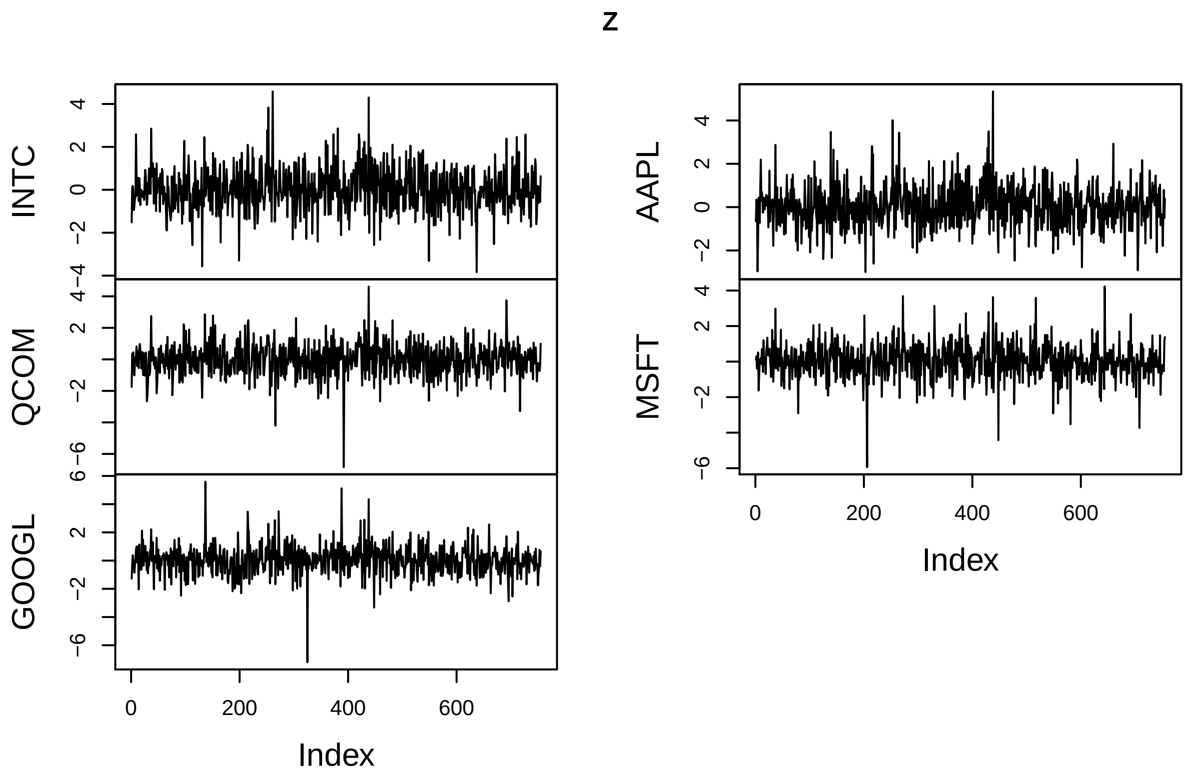
```
## Fit marginal time series
uspec <- rep(list(ugarchspec(distribution.model = "std")), ncol(X))
fit.ARMA.GARCH <- fit_ARMA_GARCH(X, ugarchspec.list = uspec)
```

```
## ===============================================================================
```

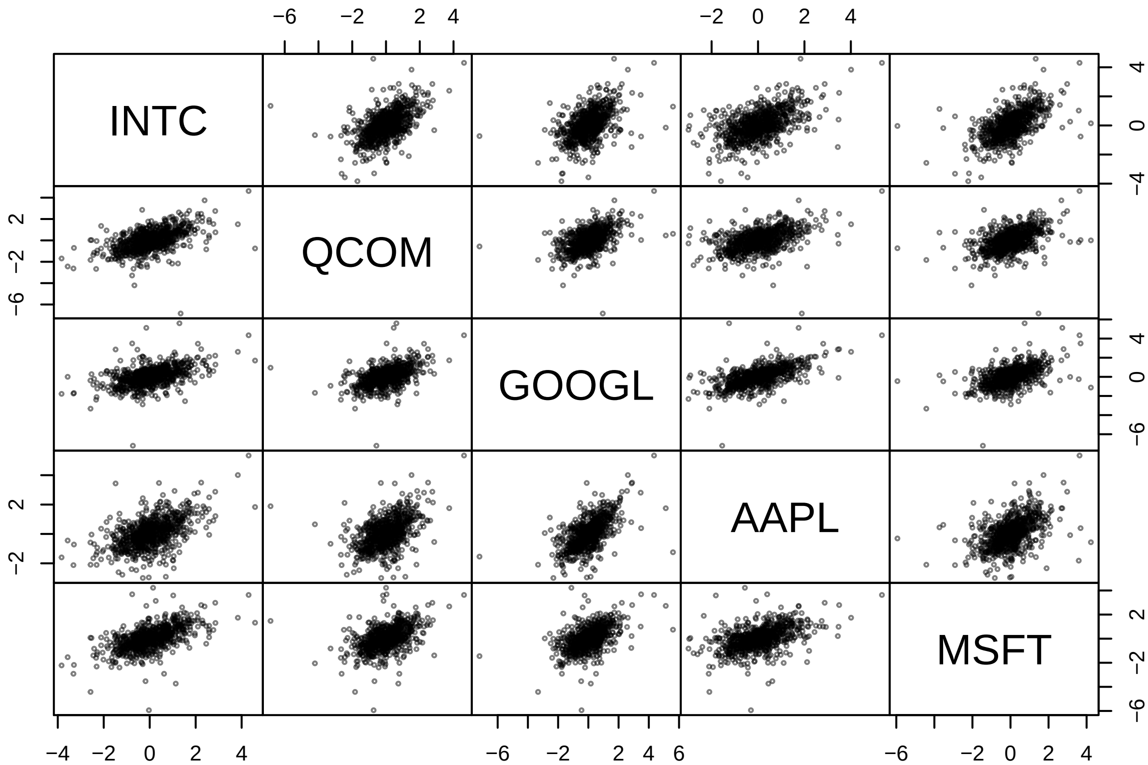
```
# fitted models
fits <- fit.ARMA.GARCH$fit
```

Next we compute the standardized residuals for each margin, and plot these

```
# a matrix n x p containing the standardized marginal residuals
Z <- as.matrix(do.call(merge, lapply(fits, residuals, standardize = TRUE))) # grab out standardized res
colnames(Z) <- colnames(S)

# visualize the standardized residuals
plot.zoo(Z)
```




```r
pairs2(Z,cex = 0.4, col = adjustcolor("black", alpha.f = 0.5))
```



And then extract the estimated degrees of freedom for each margin $\hat{\nu}_1, \ldots, \hat{\nu}_5$.

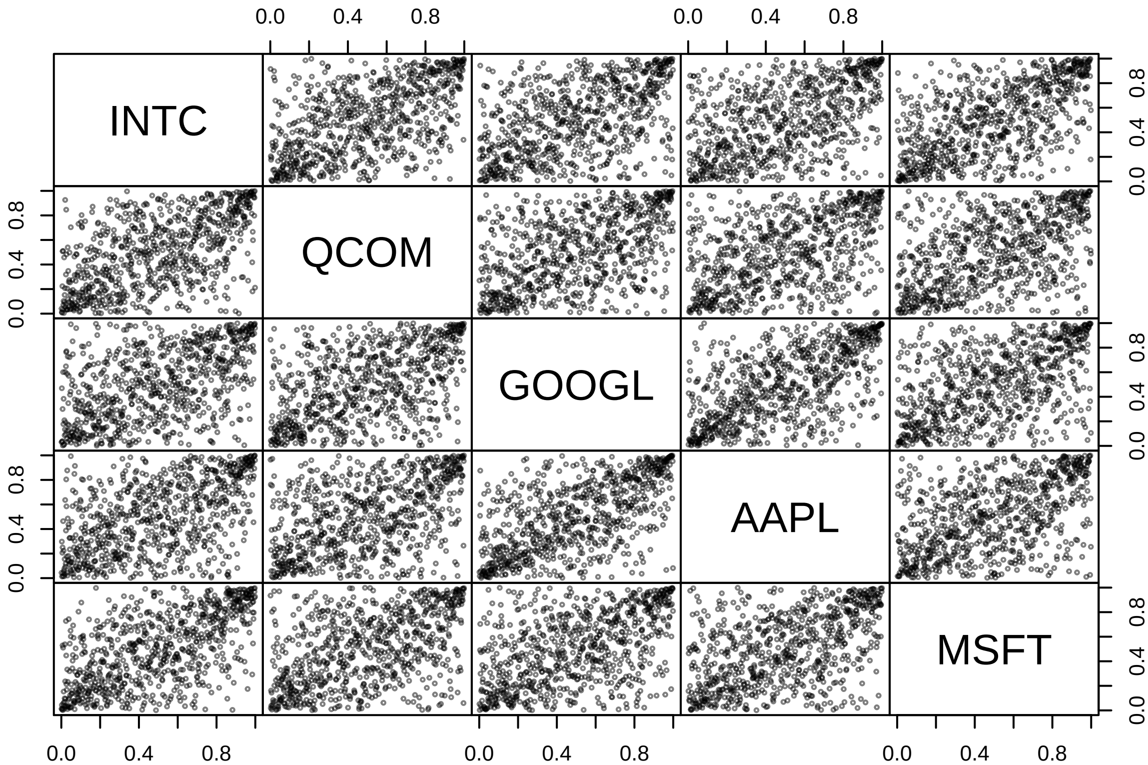
```
# vector of estimated df
(nu.mar <- vapply(fits, function(x) x@fit$coef[["shape"]], NA_real_))
```

```
## [1] 6.093672 7.151326 4.822795 7.508152 4.532731
```

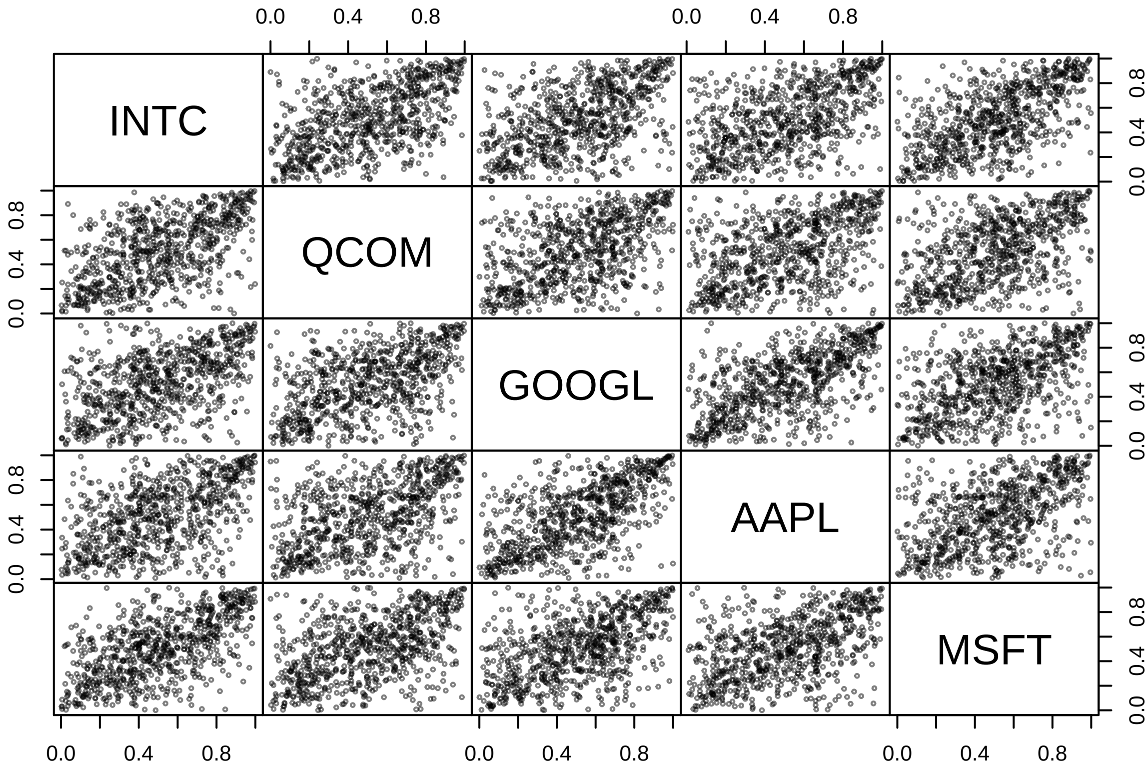
3) Fitting copulas

Pseudo-observations from the standardized t residuals are then generated, and plotted, through two methods; non-parametrically (based on the ranks) or parametrically (based on the estimated marginal distribution of the residuals, $t_{\hat{\nu}_i}$).

```
## Compute pseudo-observations from the standardized t residuals
U <- pobs(Z)
pairs2(U, cex = 0.4, col = adjustcolor("black", alpha.f = 0.5))
```




```r
# alternatively
U2<- pt(Z,nu.mar)
pairs2(U2, cex = 0.4, col = adjustcolor("black", alpha.f = 0.5))
```



Now, a series of different copulas will be fitted through the use of the fitCopula function from the **copula** package.

Starting with a Gumbel copula on the pseudo-observations.

```
## Fitting a Gumbel copula
fit.gc <- fitCopula(gumbelCopula(dim = p),
                    data = U, method = "mpl")

# print estimated copula parameter
fit.gc@estimate
```

```
## [1] 1.454779
```

```
# fitted copula
gc <- fit.gc@copula
```

The matrices of pairwise Kendall's tau and upper tail-dependence coefficients (which are both the same for all non-diagonal elements, which follows from the fact that the Gumbel copula is archimedian) based on the fitted copula are then the following:

```
## Compute matrices of pairwise Kendall's tau and upper tail-dependence coefficients
p2P(tau(gc), d = p)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.3126104 0.3126104 0.3126104 0.3126104
## [2,] 0.3126104 1.0000000 0.3126104 0.3126104 0.3126104
## [3,] 0.3126104 0.3126104 1.0000000 0.3126104 0.3126104
## [4,] 0.3126104 0.3126104 0.3126104 1.0000000 0.3126104
```

```
## [5,] 0.3126104 0.3126104 0.3126104 0.3126104 1.0000000
```

```
p2P(lambda(gc)["upper"], d = p)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.3896329 0.3896329 0.3896329 0.3896329
## [2,] 0.3896329 1.0000000 0.3896329 0.3896329 0.3896329
## [3,] 0.3896329 0.3896329 1.0000000 0.3896329 0.3896329
## [4,] 0.3896329 0.3896329 0.3896329 1.0000000 0.3896329
## [5,] 0.3896329 0.3896329 0.3896329 0.3896329 1.0000000
```

Next, the same procedure follows for a t copula. Note that the tau and tail dependence coefficients are no longer all the same this time.

```
## Fitting a t copula
fit.tc <- fitCopula(tCopula(dim = p, dispstr = "un"),
                    data = U, method = "itau.mpl")
# estimated degrees of freedom nu
(nu <- tail(fit.tc@estimate, n = 1))
```

```
## [1] 7.269678
```

```
# estimated correlation matrix
(P <- p2P(head(fit.tc@estimate, n = -1)))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.5761937 0.5246273 0.5361455 0.6246738
## [2,] 0.5761937 1.0000000 0.5229914 0.5064793 0.5123869
## [3,] 0.5246273 0.5229914 1.0000000 0.6429929 0.5391979
## [4,] 0.5361455 0.5064793 0.6429929 1.0000000 0.5249092
## [5,] 0.6246738 0.5123869 0.5391979 0.5249092 1.0000000
```

```
# fitted copula
tc <- fit.tc@copula

## Compute matrices of pairwise Kendall's tau and upper tail-dependence coefficients
p2P(tau(tc))
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.3909252 0.3515906 0.3602403 0.4295361
## [2,] 0.3909252 1.0000000 0.3503680 0.3381067 0.3424772
## [3,] 0.3515906 0.3503680 1.0000000 0.4446150 0.3625450
## [4,] 0.3602403 0.3381067 0.4446150 1.0000000 0.3518014
## [5,] 0.4295361 0.3424772 0.3625450 0.3518014 1.0000000
```

```
p2P(lambda(tc)[(choose(p,2)+1):(p*(p-1))])
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.1730450 0.1457634 0.1514811 0.2030989
## [2,] 0.1730450 1.0000000 0.1449678 0.1371596 0.1399074
## [3,] 0.1457634 0.1449678 1.0000000 0.2157428 0.1530311
## [4,] 0.1514811 0.1371596 0.2157428 1.0000000 0.1459009
## [5,] 0.2030989 0.1399074 0.1530311 0.1459009 1.0000000
```

4) Simulating paths from the full model

Here we will start from the fitted marginal distributions and t copula to simulate new random vectors of size m and dimension 5. We will repeat this simulation $B = 200$ times to get an idea of its variability. The simulation steps are explicited in the R code below.

```r
B <- 200
m <- ceiling(n/10) # length of the simulates paths
X.lst <- lapply(1:B, function(b) {
  ## 1) Simulate from the fitted copula
  U. <- rCopula(m, copula = tc)

  ## 2) Quantile-transform to standardized t distributions (for ugarchsim())
  Z. <- sapply(1:p, function(j) sqrt((nu.mar[j]-2)/nu.mar[j]) * qt(U.[,j], df = nu.mar[j]))

  ## 3) Use these multivariate dependent t innovations to sample from the time series
  sim <- lapply(1:p, function(j)
    ugarchsim(fits[[j]], n.sim = m, m.sim = 1,
              custom.dist = list(name = "sample",
                                 distfit = Z.[,j, drop = FALSE])))

  ## 4) Extract simulated series
  sapply(sim, function(x) fitted(x)) # simulated multivariate series X_t x@simulation$seriesSim
})
```

Which gererates a List of length B containing (n x p)-matrices

5) Predict the aggregated loss and $\mathrm{VaR}_{0.99}$

5.1) Compute the aggregated loss by summing up the losses of all $p = 5$ constituants at each time t in the original data X.

5.2) Compute the aggregated losses for each simulated matrix in the list 'X.lst' : We thus obtain B instances of the simulated losses over time.

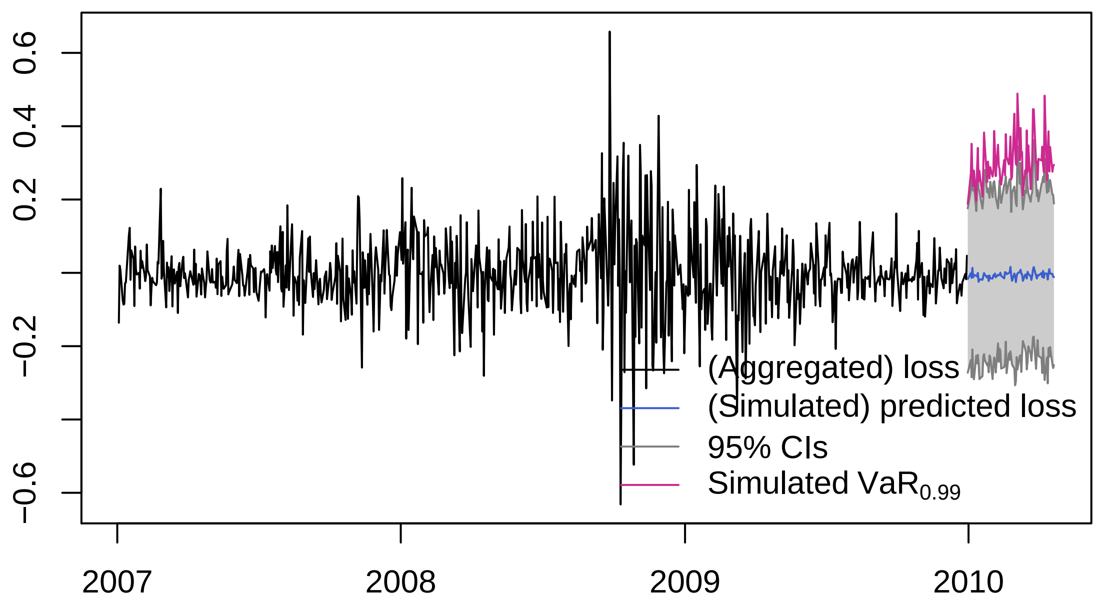
The predicted loss is the average over all simulated losses. Compute an α-confidence interval by taking the $\alpha/2$ and $1-\alpha/2$ empirical quantiles of the simulated losses.

5.3) Compute the $\mathrm{VaR}_{0.99}$ based on the simulated losses, and plot them together with the original return data.

```r
# aggregated loss; n-vector
Xs <- rowSums(X)

# simulated aggregated losses; (m, B)-matrix
Xs. <- sapply(X.lst, rowSums)

# predicted aggregated loss; m-vector
Xs.mean <- rowMeans(Xs.)

# CIs; (2, m)-matrix
Xs.CI <- apply(Xs., 1, function(x) quantile(x, probs = c(0.025, 0.975)))

# confidence level
alpha <- 0.99

# VaR_alpha; m-vector
VaR <- apply(Xs., 1, function(x) quantile(x, probs = alpha))
```

```r
tm <- index(SP500_const)
start <- match(time[1], as.character(tm))
past <- tm[start:(start+n-1)]
```

```r
future <- tm[(start+n):(start+n+m-1)]
plot(past, Xs, type = "l", xlim = range(c(past, future)), xlab = "", ylab = "") # actual (past) losses
polygon(c(future, rev(future)), c(Xs.CI[1,], rev(Xs.CI[2,])),
        border = NA, col = "grey80") # CI region
lines(future, Xs.mean, col = "royalblue3") # predicted aggregated loss
lines(future, Xs.CI[1,], col = "grey50") # lower CI
lines(future, Xs.CI[2,], col = "grey50") # upper CI
lines(future, VaR, col = "maroon3") # VaR_alpha
legend("bottomright", bty = "n", lty = rep(1, 4),
       col = c("black", "royalblue3", "grey50", "maroon3"),
       legend = c("(Aggregated) loss", "(Simulated) predicted loss",
                  "95% CIs", as.expression(substitute("Simulated"~VaR[a], list(a = alpha)))))
```



References This tutorial is bases on R code from QRMtutorials.org