Homework 3

Vladislav Zakatov

23 October 2015

This document has been created as part of the third homework assignment at CMF.

Initially, the working directory, system locale are set and the required packages are loaded.

```
##### Initialisation #####
setwd("~/")
library(quantmod)
library(ggplot2)
library(ghyp)
library(copula)
Sys.setlocale("LC_ALL","English")
```

The historical prices of Apple and Google from 2014 up to present are loaded from Yahoo Finance using *getSymbols* function from *quantmod* package.

```
# specify the tickers to load
tickers = c("AAPL","GOOGL")

# environment where data will be stored
e <- new.env()

# suppress warning about future change in the function behaviour
options("getSymbols.warning4.0"=FALSE)

# load data from Yahoo Finance
getSymbols(tickers, src = "yahoo", from="2014-01-01", env=e)

# merge data into a single xts taking only adjusted prices
data = do.call(merge, eapply(e, Ad)[tickers])
names(data) = tickers
```

Daily returns are calculated and saved as *xts*.

```
##### Calculate returns #####
rets = apply(data, 2, function(x) diff(x)/x[-length(x)])
rets = xts(rets, order.by = index(data)[-1])
```

We initiate our portfolio consisting of Apple and Google returns After that we fit the multivariate generalized hyperbolic distribution to the returns and calculate value at risk as well as expected shortfall for the whole period (2014 – present). For this calculation we assume equal weights of the assets in portfolio.

```
##### Initiate portfolio #####
prt = rets

##### Fit multivariate GHD and assess portfolio risks #####
prt.fit <- fit.ghypmv(prt,symmetric=FALSE,silent=TRUE)
```

```
w <- c(0.5,0.5) # assets' weights in portfolio
N <- 10^6; alpha <- 0.1
sim <- rghyp(n=N,object=prt.fit)
prt.sim <- w[1]*sim[,1]+w[2]*sim[,2]
prt.sim <- sort(prt.sim)
VaR <- prt.sim[alpha*N]
ES <- mean(prt.sim[1:(alpha*N-1)])

print(VaR)
```

```
## [1] -0.01322974
```

```
print(ES)
```

```
## [1] -0.02169165
```

The estimated value at risk is about -1.325% and expected shortfall is equal to -2.1676%.

We now want to optimize out portfolio weight with respect to the minimal value at risk with 95% confidence.

```
##### Find optimal portfolio weights and assess portfolio risks #####
opt <- portfolio.optimize(prt.fit,
                          risk.measure="value.at.risk",type="minimum.risk",
                          target.return=NULL,risk.free=NULL,level=0.95,silent=TRUE)

w <- opt$opt.weights # assets' weights in portfolio
N <- 10^6; alpha <- 0.1
sim <- rghyp(n=N,object=prt.fit)
prt.sim <- w[1]*sim[,1]+w[2]*sim[,2]
prt.sim <- sort(prt.sim)
VaR_new <- prt.sim[alpha*N]
ES_new <- mean(prt.sim[1:(alpha*N-1)])

print(VaR_new)
```

```
## [1] -0.0132613
```

```
print(ES_new)
```

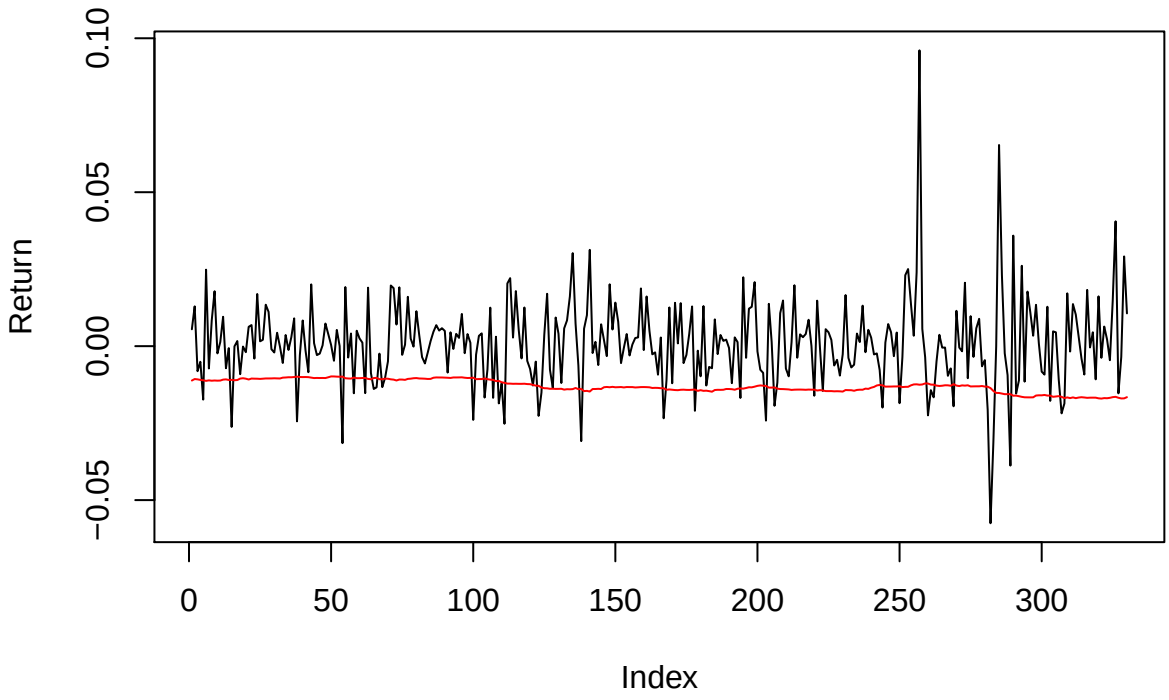
```
## [1] -0.02165863
```

The new portfolio weights appear to be pretty close to the default ones. Value at risk estimated for the new portfolio is about -1.324% and expected shortfall is equal to -2.1667%, which is slightly lower that for the default portfolio weights.

We now calculate estimate the VaR curve with window of length 130 trading days. We also choose to readjust the weights of the assets in the portfolio on every step basing on the last 130 days of data while minimzing value at risk.

```
##### Calculate VaR curve #####
VaR_curve <- numeric()
h <- 0.5 * 260 # window length
h.w <- matrix(ncol = 2, nrow = length(prt[,1]) - h)

for (i in (h+1):length(prt[,1]))
{
  h.prt <- prt[(i-h):(i-1),]

  h.prt.fit <- fit.ghypmv(h.prt,symmetric=FALSE,silent=TRUE)

  h.opt <- portfolio.optimize(h.prt.fit,
                              risk.measure="value.at.risk",type="minimum.risk",
                              target.return=NULL,risk.free=NULL,level=0.95,silent=TRUE)

  h.w[i-h,] <- h.opt$opt.weights # assets' weights in portfolio

  sim <- rghyp(n=N,object=h.prt.fit)
  h.prt.sim <- h.w[i-h,1]*sim[,1]+h.w[i-h,2]*sim[,2]
  h.prt.sim <- sort(h.prt.sim)
  VaR_curve[i-h] <- h.prt.sim[alpha*N]
}
```

```
fact <- prt[(h+1):length(prt[,1]),1]*h.w[,1] + prt[(h+1):length(prt[,2]),2]*h.w[,2]
plot(as.numeric(fact),type="l", ylab = "Return")
lines(VaR_curve,col="red")
```



The graph shows that the resulting VaR curve is almost straight, which provides evidence that our calculations of VaR were correctly modelled and reliable

To prove the relevance of the VaR curve we use the Kupiec test to check whether the number of actual returns lower than VaR is not greater than it is to be expected.

```
##### Kupiec test #####
K <- sum(fact<VaR_curve); alpha0 <- K/length(rets)
S <- -2*log((1-alpha)^(length(rets)-K)*alpha^K)+
  2*log((1-alpha0)^(length(rets)-K)*alpha0^K)
p.value <- 1-pchisq(S,df=1)

print(alpha)
```

```
## [1] 0.1
```

```
print(alpha0)
```

```
## [1] 0.0423913
```

```
print(p.value)
```

```
## [1] 7.512935e-11
```

While Kupiec test rejects the null hypothesis with a very low p-value, we ascert that to the fact that we re-optimzed the weights of assets in the portfolio on every step. And hence the actual number of returns (4.24%) is lower that had been projected (10%), which seems to be a rather good result.

We now move on to modelling portfolio returns with copulas. We start with fitting best distributions from GHD class based on AIC criterion. After that we construct a vector with probability distributions of both assets.

```
##### Fit assets' distributions #####
aapl.fit <- stepAIC.ghyp(rets$AAPL,dist=c("gauss","t","ghyp"),
                         symmetric=NULL,silent=TRUE)$best.model
```

```
## Currently fitting: asymmetric t
## Currently fitting: asymmetric ghyp
## Currently fitting: symmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: gauss
```

```
googl.fit <- stepAIC.ghyp(rets$GOOGL,dist=c("gauss","t","ghyp"),
                         symmetric=NULL,silent=TRUE)$best.model
```

```
## Currently fitting: asymmetric t
## Currently fitting: asymmetric ghyp
## Currently fitting: symmetric t
## Currently fitting: symmetric ghyp
## Currently fitting: gauss
```

```
aapl.cdf <- pghyp(rets$AAPL,object=aapl.fit)
googl.cdf <- pghyp(rets$GOOGL,object=googl.fit)
cdf <- cbind(as.numeric(aapl.cdf), as.numeric(googl.cdf))
```

We initialise four different copulas and fit them to our data.

```
##### Initialise copulas #####
norm.cop <- normalCopula(dim=2,param=0.5,dispstr="un")
stud.cop <- tCopula(dim=2,param=0.5,df=5,df.fixed=TRUE,dispstr="un")
gumb.cop <- gumbelCopula(dim=2,param=2)
clay.cop <- claytonCopula(dim=2,param=2)

norm.fit <- fitCopula(cdf,copula=norm.cop)
stud.fit <- fitCopula(cdf,copula=stud.cop)
gumb.fit <- fitCopula(cdf,copula=gumb.cop)
clay.fit <- fitCopula(cdf,copula=clay.cop)
```

We now simulate portfolio returs assuming equal weights.

```
N <- 10^4
stud.sim <- rCopula(n=N,copula=stud.fit@copula)

aapl.sim <- qghyp(stud.sim[,1],object=aapl.fit)
googl.sim <- qghyp(stud.sim[,2],object=googl.fit)
w_copula <- c(0.5,0.5)
prt.sim <- w_copula[1]*aapl.sim + w_copula[2]*googl.sim
```

Finally, we calculate value at risk and expected shortfall for the whole period.

```
##### VaR and ES #####
alpha <- 0.10
prt.sim <- sort(prt.sim)
VaR <- prt.sim[alpha*N]
ES <- mean(prt.sim[1:(alpha*N-1)])

print(VaR)
```

```
## [1] -0.01377468
```

```
print(ES)
```

```
## [1] -0.02296604
```

Estimation of VaR curve for copulas. Still running... Z-z-z...

```
##### Calculate VaR curve #####
VaR_curve <- numeric()
h <- 0.5 * 260 # window length


for (i in (h+1):length(rets[,1]))
{
  h.rets <- rets[(i-h):(i-1),]

  aapl.fit <- stepAIC.ghyp(h.rets$AAPL,dist=c("t","ghyp"),
                           symmetric=NULL,silent=TRUE)$best.model
  googl.fit <- stepAIC.ghyp(h.rets$GOOGL,dist=c("t","ghyp"),
                            symmetric=NULL,silent=TRUE)$best.model

  aapl.cdf <- pghyp(h.rets$AAPL,object=aapl.fit)
  googl.cdf <- pghyp(h.rets$GOOGL,object=googl.fit)
  cdf <- cbind(as.numeric(aapl.cdf), as.numeric(googl.cdf))

  stud.fit <- fitCopula(cdf,copula=stud.cop)

  aapl.sim <- qghyp(stud.sim[,1],object=aapl.fit)
  googl.sim <- qghyp(stud.sim[,2],object=googl.fit)

  h.prt.sim <- w[1]*aapl.sim + w[2]*googl.sim

  h.prt.sim <- sort(h.prt.sim)
  VaR_curve[i-h] <- h.prt.sim[alpha*N]
}

fact <- prt[(h+1):length(prt[,1]),1]*h.w[,1] + prt[(h+1):length(prt[,2]),2]*h.w[,2]
plot(as.numeric(fact),type="l", ylab = "Return")
lines(VaR_curve,col="red")

##### Kupiec test #####
```

```
K <- sum(fact<VaR_curve); alpha0 <- K/length(rets)
S <- -2*log((1-alpha)^(length(rets)-K)*alpha^K)+
  2*log((1-alpha0)^(length(rets)-K)*alpha0^K)
p.value <- 1-pchisq(S,df=1)
```