```
# Script file developed by

# M. Concepción Ausín
# Cristina G. de la Fuente
# Aitor J. Farragut

# This document was created within the framework of a final degree project
# presented by Aitor Juan Farragut in order to obtain the Bachelor's degree
# in Economics at Universidad Carlos III de Madrid.

# The name of the project is
# "Estimación de series temporales financieras multivariantes con modelos de
# cópulas."

# May, 2017

# Clean workspace
rm(list = ls())

############################################################################
# We start generating d time series from a GARCH(1,1) process with innova-
# tions whose dependance structure is modeled using copulas.
############################################################################

# Load the necessary libraries
library(copula)
library(parallel)
library(rugarch)
library(CDVine)

# samle size
T = 1000
# dimension
d = 4

fam = c(5, 2, 3, 6, 4, 2)
# copula 12 Frank
# copula 13 t
# copula 14 Clayton
# copula 23|1 Joe
# copula 24|1 Gumbel
# copula 34|12 t

par = c(-1.3, 0.7, 1.5, 2, 4.2, -0.3)
# \theta Frank, \rho t, \theta Clayton, \theta Joe, \theta Gumbel, \rho t
par2 = c(0, 3, 0, 0, 0, 5)
# null Frank, \eta t, null Clayton, null Joe, null Gumbel, \eta t

# Establish the random seed that will allow us to reproduce the same data
# each time.
set.seed(777)
```

```
# Generate random observations of U according to the model.
U = CDVineSim(T, family = fam, par = par, par2 = par2, type = 1)
pairs(U, labels = c("U_1", "U_2", "U_3", "U_4"), main = "Simuladas", col =
"royalblue4")

tau_pairs_U = cor(U, method = "k")

tau_cop = matrix(nrow = (d*(d-1)/2), ncol = 1)
for(i in 1:(d*(d-1)/2))
{
	tau_cop[i] = BiCopPar2Tau(fam[i], par = par[i], par2 = par2[i])
}

# Marginal t degrees of freedom.
nu = c(7, 4, 5, 2.5)

X = matrix(nrow = T, ncol = d)
for(i in 1:d)
{
	X[,i] = qt(U[,i], df = nu[i])
}

# The innovations must have 0 mean and unit variance.
# They already have 0 mean (the t distribution is centered), but we
# standardize to obtain the unit variance.

epsilon = matrix(nrow = T, ncol = d)
for(i in 1:d)
{
	epsilon[,i] = sqrt((nu[i] - 2)/nu[i])* X[,i]
}

# Plot the simulated t-distributed innovations.
par(mfrow = c(d,1))
for(i in 1:d)
{
	plot(1:T, epsilon[,i], "l", col = "blue", xlab = "t", ylab =
paste("Innovaciones serie", i))
}

# Fix the parameters for the marginal time models.

mu = c(0.1, -0.1, 0.3, 0.05)
omega = c(0.2, 0.7, 0.3, 0.9)
alpha = c(0.25, 0.15, 0.17, 0.08)
beta = c(0.7, 0.8, 0.75, 0.9)

# With the defined parameters and the simulated innovations, we use the
# corresponding dependence structure and generate the d random walks.

h = matrix(nrow = T, ncol = d)
Y = matrix(nrow = T, ncol = d)
```

```
h[1,] = omega
for(i in 1:d)
{
        Y[1,i] = mu[i] + sqrt(h[1,i]) * epsilon[1,i]
}

for(k in 2:T)
{
        for(i in 1:d)
        {
                h[k,i] = omega[i] + alpha[i]*( Y[(k-1),i] - mu[i] )^2 + 
beta[i]*h[(k-1),i]
                Y[k,i] = mu[i] + sqrt(h[k,i]) * epsilon[k,i]
        }
}

# Plot the marginal time series
par(mfrow = c(d,1))
for(i in 1:d)
{
        matplot(Y[,i], type = "l", xlab = "t", ylab = paste("Serie", i), col = 
"blue")
}


for(i in 1:d)
{
        matplot(Y[,i], type = "l", xlab = "t", ylab = paste("Serie", i), col = 
"blue")
#       abline(h=0)
        readline(prompt="Press [enter] to continue")
}

ymin = min(Y) - 1
ymax = max(Y) + 1
matplot(Y, type = "l", xlab = "t", ylab = "Series solapadas", ylim=c(ymin, 
ymax), lwd = 2, col = c(1:d), lty = 5)
legend(130, -12.5, c("Serie 1", "Serie 2", "Serie 3", "Serie 4"), lty = 5, col = 
1:d, lwd = 2)

###########################################################################
# Now we estimate the model parameters for the simulated data.
###########################################################################

# Given the multidimensional time series, Y (Txd), we adjust a copula-GARCH(1,1)
# process in d dimensions. The innovations, marginally, follow a Student's-t
# distribution and their relation is modeled by means of a vine copula.

meanModel = list(armaOrder = c(0,0), include.mean=TRUE)
varModel = list(model = "sGARCH", garchOrder = c(1,1)) # standard GARCH
```

```
uspec = ugarchspec(varModel, mean.model = meanModel, distribution.model = "std")

fit = NULL

for(i in 1:d)
{
	fit = cbind(fit, ugarchfit(data = Y[,i], spec = uspec))
}

mu_hat = matrix(nrow = d, ncol = 1)
omega_hat = matrix(nrow = d, ncol = 1)
alpha_hat = matrix(nrow = d, ncol = 1)
beta_hat = matrix(nrow = d, ncol = 1)
nu_hat = matrix(nrow = d, ncol = 1)

for(i in 1:d)
{
	mu_hat[i] = fit[[i]]@fit$coef[1]
	omega_hat[i] = fit[[i]]@fit$coef[2]
	alpha_hat[i] = fit[[i]]@fit$coef[3]
	beta_hat[i] = fit[[i]]@fit$coef[4]
	nu_hat[i] = fit[[i]]@fit$coef[5]
}

epsilon_hat = sapply(fit, residuals, standardize = TRUE)
U_hat = pobs(epsilon_hat)
pairs(U_hat, labels = c("U_1", "U_2", "U_3", "U_4"), main = "Estimadas", col =
"royalblue4")

tau_pairs_U_hat = cor(U_hat, method = "k")

# In order to estimate the parameters related to the joint behavior of
# the innovations, we begin by estimating the copula parameters for all
# the possible orders of the variables.

fam_1234 = CDVineCopSelect(U_hat, type=1, familyset = 1:6, selectioncrit =
"AIC", indeptest = T)$family
Cop_hat_1234 = CDVineMLE(U_hat, type = 1, family = fam_1234)

fam_1324 = CDVineCopSelect(U_hat[,c(1,3,2,4)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_1324 = CDVineMLE(U_hat[,c(1,3,2,4)], type = 1, family = fam_1324)

fam_1432 = CDVineCopSelect(U_hat[,c(1,4,3,2)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_1432 = CDVineMLE(U_hat[,c(1,4,3,2)], type = 1, family = fam_1432)

fam_2134 = CDVineCopSelect(U_hat[,c(2,1,3,4)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_2134 = CDVineMLE(U_hat[,c(2,1,3,4)], type = 1, family = fam_2134)

fam_2314 = CDVineCopSelect(U_hat[,c(2,3,1,4)], type=1, familyset = 1:6,
```

```
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_2314 = CDVineMLE(U_hat[,c(2,3,1,4)], type = 1, family = fam_2314)

fam_2413 = CDVineCopSelect(U_hat[,c(2,4,1,3)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_2413 = CDVineMLE(U_hat[,c(2,4,1,3)], type = 1, family = fam_2413)

fam_3124 = CDVineCopSelect(U_hat[,c(3,1,2,4)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_3124 = CDVineMLE(U_hat[,c(3,1,2,4)], type = 1, family = fam_3124)

fam_3214 = CDVineCopSelect(U_hat[,c(3,2,1,4)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_3214 = CDVineMLE(U_hat[,c(3,2,1,4)], type = 1, family = fam_3214)

fam_3412 = CDVineCopSelect(U_hat[,c(3,4,1,2)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_3412 = CDVineMLE(U_hat[,c(3,4,1,2)], type = 1, family = fam_3412)

fam_4123 = CDVineCopSelect(U_hat[,c(4,1,2,3)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_4123 = CDVineMLE(U_hat[,c(4,1,2,3)], type = 1, family = fam_4123)

fam_4231 = CDVineCopSelect(U_hat[,c(4,2,3,1)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_4231 = CDVineMLE(U_hat[,c(4,2,3,1)], type = 1, family = fam_4231)

fam_4312 = CDVineCopSelect(U_hat[,c(4,3,1,2)], type=1, familyset = 1:6,
selectioncrit = "AIC", indeptest = T)$family
Cop_hat_4312 = CDVineMLE(U_hat[,c(4,3,1,2)], type = 1, family = fam_4312)

# We check the value of the AIC for each order.

AIC_1234 = CDVineAIC(U_hat, family = fam, par = Cop_hat_1234$par, par2 =
Cop_hat_1234$par2, type = 1)$AIC
AIC_1324 = CDVineAIC(U_hat[,c(1,3,2,4)], family = fam_1324, par =
Cop_hat_1324$par, par2 = Cop_hat_1324$par2, type = 1)$AIC
AIC_1432 = CDVineAIC(U_hat[,c(1,4,3,2)], family = fam_1432, par =
Cop_hat_1432$par, par2 = Cop_hat_1432$par2, type = 1)$AIC
AIC_2134 = CDVineAIC(U_hat[,c(2,1,3,4)], family = fam_2134, par =
Cop_hat_2134$par, par2 = Cop_hat_2134$par2, type = 1)$AIC
AIC_2314 = CDVineAIC(U_hat[,c(2,3,1,4)], family = fam_2314, par =
Cop_hat_2314$par, par2 = Cop_hat_2314$par2, type = 1)$AIC
AIC_2413 = CDVineAIC(U_hat[,c(2,4,1,3)], family = fam_2413, par =
Cop_hat_2413$par, par2 = Cop_hat_2413$par2, type = 1)$AIC
AIC_3124 = CDVineAIC(U_hat[,c(3,1,2,4)], family = fam_3124, par =
Cop_hat_3124$par, par2 = Cop_hat_3124$par2, type = 1)$AIC
AIC_3214 = CDVineAIC(U_hat[,c(3,2,1,4)], family = fam_3214, par =
Cop_hat_3214$par, par2 = Cop_hat_3214$par2, type = 1)$AIC
AIC_3412 = CDVineAIC(U_hat[,c(3,4,1,2)], family = fam_3412, par =
Cop_hat_3412$par, par2 = Cop_hat_3412$par2, type = 1)$AIC
AIC_4123 = CDVineAIC(U_hat[,c(4,1,2,3)], family = fam_4123, par =
```

```
Cop_hat_4123$par, par2 = Cop_hat_4123$par2, type = 1)$AIC
AIC_4231 = CDVineAIC(U_hat[,c(4,2,3,1)], family = fam_4231, par =
Cop_hat_4231$par, par2 = Cop_hat_4231$par2, type = 1)$AIC
AIC_4312 = CDVineAIC(U_hat[,c(4,3,1,2)], family = fam_4312, par =
Cop_hat_4312$par, par2 = Cop_hat_4312$par2, type = 1)$AIC

AIC_1234
AIC_1324
AIC_1432
AIC_2134
AIC_2314
AIC_2413
AIC_3124
AIC_3214
AIC_3412
AIC_4123
AIC_4231
AIC_4312

min(c(AIC_1234, AIC_1324, AIC_1432, AIC_2134, AIC_2314, AIC_2413, AIC_3124,
AIC_3214, AIC_3412, AIC_4123, AIC_4231, AIC_4312))

# We order U_hat according to the value of the corresponding AIC.
# In this case, and with this criterion, we choose 1, 2, 3, 4, which is
# the order we would have wanted to choose, knowing that we are choosing
# the original order in which the data was simulated.

fam_hat = fam_1234
# The copula family selected by the chosen order matches the one used
# during the simulations, which is a good sign.
Cop_hat = Cop_hat_1234

# Save the estimated parameters
# Be careful with the names if the selected order changes!

par_hat = Cop_hat$par
par2_hat = Cop_hat$par2

tau_cop_hat = matrix(nrow = (d*(d-1)/2), ncol = 1)
for(i in 1:(d*(d-1)/2))
{
        tau_cop_hat[i] = BiCopPar2Tau(fam_hat[i], par = par_hat[i], par2 =
par2_hat[i])
}

# fitted vs real marginal parameters.
realVSest_marginal = matrix(nrow = d, ncol = 10)
for(i in 1:d)
{
        realVSest_marginal[i,] = c(mu[i], mu_hat[i], omega[i], omega_hat[i],
alpha[i], alpha_hat[i], beta[i], beta_hat[i], nu[i], nu_hat[i])
}
```

```
colnames(realVSest_marginal) = c("mu", "mu_hat", "omega", "omega_hat", "alpha",
"alpha_hat", "beta", "beta_hat", "nu", "nu_hat")
nombres_realVSest_marginal = NULL
for(i in 1:d)
{
	nombres_realVSest_marginal = cbind(nombres_realVSest_marginal,
paste("serie", i))
}
rownames(realVSest_marginal) = nombres_realVSest_marginal

realVSest_marginal

# fitted vs real parameters of the copula.
realVSest_cop = matrix(nrow = (d*(d-1)/2), ncol = 6)

realVSest_cop = cbind(par, par_hat, par2, par2_hat, tau_cop, tau_cop_hat)
colnames(realVSest_cop) = c("par", "par_hat", "par2", "par2_hat", "tau",
"tau_hat")
rownames(realVSest_cop) = c("copula 12: Frank", "copula 13: t", "copula 14:
Clayton", "copula 23|1: Joe", "copula 24|1: Gumbel", "copula 34|12: t")

realVSest_cop

realVSest_corr = cbind(tau_pairs_U, tau_pairs_U_hat)

realVSest_corr

###########################################################################
# Now we evaluate the quality of our fittings using volatility plots.
###########################################################################

# We will plot the real and estimated volatilities in order to be able to
# compare them.

# First, we compute the estimated volatilities using the estimated innovations
# and the estimated values of the GARCH parameters.

h_hat = matrix(nrow = T, ncol = d)

h_hat[1,] = omega_hat

for(k in 2:T)
{
	for(i in 1:d)
	{
		h_hat[k,i] = omega_hat[i] + alpha_hat[i]*( Y[(k-1),i] -
mu_hat[i] )^2 + beta_hat[i]*h_hat[(k-1),i]
	}
}

# Let's compare, for each series, the real and estimated volatilities.
# First, we do it with a different plot for each series.
```

```
for(i in 1:d)
{
        H = cbind(h[,i], h_hat[,i])
        pos = max(H)
        # Overlaped plot of the simulated and estimated volatilities of the i-th
series.
        matplot(H, type = "l", xlab = "t", ylab = "h_it", col = c("red",
"blue"), main = paste("Volatilidades serie", i))
        legend(425, pos, c("Real", "Estimada"), col = c("red", "blue"), lty =
1:2, lwd = 1)
        readline(prompt="Press [enter] to continue")
}

# Now we put all d plots in a single window.
par(mfrow = c(d,1))
for(i in 1:d)
{
        H = cbind(h[,i], h_hat[,i])
        pos = max(H)
        # Overlaped plot of the simulated and estimated volatilities of the i-th
series.
        matplot(H, type = "l", xlab = "t", ylab = "h_it", col = c("red",
"blue"), main = paste("Volatilidades serie", i))
        legend(425, pos, c("Real", "Estimada"), col = c("red", "blue"), lty =
1:2, lwd = 1)
}

############################################################################
# Now, let us compute the predictive VaR for time T+1.
############################################################################

alphaVaR = 0.05

# First, we use the real parameters and volatilities to compute the
# individual VaR of each series (just to get to know each series a bit).

VaR_real = matrix(nrow = d, ncol = 1)

for(i in 1:d)
{
        VaR_real[i] = -(mu[i] + sqrt(h[T,i])*qt(alphaVaR, df = nu[i], lower.tail
= TRUE))
}

# Second, we use the estimated parameters and volatilities to compute
# the individual estimated VaR of the series under study.

VaR_hat = matrix(nrow = d, ncol = 1)

for(i in 1:d)
{
        VaR_hat[i] = -(mu_hat[i] + sqrt(h_hat[T,i])*qt(alphaVaR, df = nu_hat[i],
```

```
lower.tail = TRUE))
}

# fitted vs true predictive VaR
VaR_compare = cbind(VaR_real, VaR_hat)

colnames(VaR_compare) = c("Real", "Estimado")

VaR_compare

###########################################################################
# Given a weight vector, we compute the portfolio VaR using simulation in
# order to be able to estimate.
###########################################################################

# Number of predictive simulations.
M = 5000

# First, we estimate the VaR with the real parameters.

# Us for the innovations with the appropiate dependance.
U_VaR_R = CDVineSim(M, family = fam, par = par, par2 = par2, type = 1)

# "Observations" from the Student's-t distribution.
aux_R = matrix(nrow = M, ncol = d)
for(j in 1:M)
{
        for(i in 1:d)
        {
                aux_R[j,i] = qt(U_VaR_R[j,i], df = nu[i])
        }
}

# Simulation of the innovations for time T+1
sim_Eps_R = matrix(nrow = M, ncol = d)
for(j in 1:M)
{
        for(i in 1:d)
        {
                sim_Eps_R[j,i] = sqrt((nu[i] - 2)/nu[i]) * aux_R[j,i]
        }
}

# Computation of the simulated volatilities and simulation of
# the log-returns for time T+1
sim_h_R = matrix(nrow = M, ncol = d)
sim_Y_R = matrix(nrow = M, ncol = d)

for(j in 1:M)
{
        for(i in 1:d)
        {
```

```
                sim_h_R[j,i] = omega[i] + alpha[i]*( Y[T,i] - mu[i] )^2 +
beta[i]*h[T,i]
                sim_Y_R[j,i] = mu[i] + sqrt(sim_h_R[j,i]) * sim_Eps_R[j,i]
        }
}

# Now, we build the portfolio, given a set of weights, where
# w[i] is the weight of the i-th asset.
a_w = c(0.31, 0.16, 0.28)
w = matrix(c(a_w, (1-sum(a_w))), ncol = d, nrow = 1)

if(sum(w) != 1)
{
        print("Careful! The portfolio weights don't add up to 1")
}

# With the established weights, we will build a vector that contains
# the simulated porfolio "log-return" at time T+1.

portfolio_R = sim_Y_R %*% t(w)
hist(portfolio_R)

# With the simulations of portfolios at time T+1 (with the real parameters),
# we estimate the predictive VaR for time T+1.

VaR_R = -quantile(portfolio_R, alphaVaR)

# Once we have estimated the VaR, we can estimate the CVaR
# (Conditional VaR or expected shortfall)

subSample_R = NULL

for(j in 1:M)
{
        if(portfolio_R[j] <= (-VaR_R))
        {
                subSample_R = rbind(subSample_R, portfolio_R[j])
        }
}

CVaR_R = -mean(subSample_R)

#####
# Now, we can estimate the VaR using the estimated parameters.
#####

U_VaR_hat = CDVineSim(M, family = fam, par = par_hat, par2 = par2_hat, type = 1)

# "Observations" of the Student's-t distribution.
X_hat = matrix(nrow = M, ncol = d)
for(j in 1:M)
{
```

```
        for(i in 1:d)
        {
                X_hat[j,i] = qt(U_VaR_hat[j,i], df = nu_hat[i])
        }
}

# Simulation of the innovations for time T+1
sim_Eps_hat = matrix(nrow = M, ncol = d)
for(j in 1:M)
{
        for(i in 1:d)
        {
                sim_Eps_hat[j,i] = sqrt((nu_hat[i] - 2)/nu_hat[i]) * X_hat[j,i]
        }
}

# Computation of the simulated volatilities and simulation of
# the log-returns at time T+1.
sim_h_hat = matrix(nrow = M, ncol = d)
sim_Y_hat = matrix(nrow = M, ncol = d)

for(j in 1:M)
{
        for(i in 1:d)
        {
                sim_h_hat[j,i] = omega_hat[i] + alpha_hat[i]*( Y[T,i] -
mu_hat[i] )^2 + beta_hat[i]*h_hat[T,i]
                sim_Y_hat[j,i] = mu_hat[i] + sqrt(sim_h_hat[j,i]) *
sim_Eps_hat[j,i]
        }
}

# With the corresponding weights, we build a vector that contains the
# "log-return" of the simulated porfolio for time T+1.

portfolio_hat = sim_Y_hat %*% t(w)
hist(portfolio_hat)

# With the simulations of the portfolios for time T+1 (with the estimated
# parameters), we estimate the predictive VaR for time T+1.

VaR_hat = -quantile(portfolio_hat, alphaVaR)

# Once we have estimated the VaR, we can estimate the CVaR.

subSample_hat = NULL

for(j in 1:M)
{
        if(portfolio_hat[j] <= (-VaR_hat))
        {
                subSample_hat = rbind(subSample_hat, portfolio_hat[j])
```

```
        }
}

CVaR_hat = -mean(subSample_hat)

qqplot(portfolio_R, portfolio_hat, xlim = c(-30, 30), ylim = c(-30, 30), col =
"blue", xlab = "Portafolio simulado con parámetros reales", ylab = "Portafolio
simulado con parámetros estimados")
lines(c(-20, 20), c(-20, 20), col = "red")

# Estimated (by simulation) predictive VaR with real and estimated parameters.
VaR_port_compare = cbind(c(VaR_R, CVaR_R), c(VaR_hat, CVaR_hat))

colnames(VaR_port_compare) = c("Est x sim c/ parám reales", "Est x sim c/ parám
estimados")
rownames(VaR_port_compare) = c("VaR", "CVaR")

VaR_port_compare
```