Introduction to Monte Carlo in Finance

3 - Multi Dimensional Monte Carlo

Giovanni Della Lunga

WORKSHOP IN QUANTITATIVE FINANCE

Bologna - May 2-3, 2019

Outline

Subsection 1

The Gaussian World: Choleski Decomposition

Choleski Decomposition

- The **Choleski Decomposition** makes an appearance in Monte Carlo Methods where it is used to simulate systems with correlated variables.
- Cholesky decomposition is applied to the correlation matrix, providing a lower triangular matrix A, which when applied to a vector of uncorrelated samples, u, produces the covariance vector of the system. Thus it is highly relevant for quantitative trading.
- The standard procedure for generating a set of correlated normal random variables is through a linear combination of uncorrelated normal random variables;
- Assume we have a set of n independent standard normal random variables Z and we want to build a set of n correlated standard normals Z' with correlation matrix Σ

$$Z' = AZ, \qquad AA^t = \Sigma$$

Choleski Decomposition

- We can find a solution for A in the form of a triangular matrix

$$\begin{pmatrix} A_{11} & 0 & \dots & 0 \\ A_{21} & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}$$

- **diagonal elements**

$$a_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} a_{ik}^2}$$

- **off-diagonal elements**

$$a_{ij} = \frac{1}{a_{ii}} \left(\Sigma_{ij} - \sum_{k=1}^{i-1} a_{ik} a_{jk} \right)$$

Cholesky Decomposition

- Using Python, the most efficient method in both development and execution time is to make use of the NumPy/SciPy linear algebra (linalg) library, which has a built in method cholesky to decompose a matrix.
- The optional lower parameter allows us to determine whether a lower or upper triangular matrix is produced:

```
import pprint
import scipy
import scipy.linalg   # SciPy Linear Algebra Library

A = scipy.array([[6, 3, 4, 8],
                 [3, 6, 5, 1],
                 [4, 5, 10, 7],
                 [8, 1, 7, 25]])
L = scipy.linalg.cholesky(A, lower=True)
U = scipy.linalg.cholesky(A, lower=False)
```

Choleski Decomposition

- For example, for a two-dimension random vector we have simply

$$A = \begin{pmatrix} \sigma_1 & 0 \\ \sigma_2 \rho & \sigma_2 \sqrt{1-\rho^2} \end{pmatrix}$$

- say one needs to generate two correlated normal variables x_1 and x_2
- All one needs to do is to generate two uncorrelated Gaussian random variables z_1 and z_2 and set

$$x_1 = z_1$$

$$x_2 = \rho z_1 + \sqrt{1-\rho^2} z_2$$

Cholesky Decomposition

In Python everything you need is available in the numpy library, as we can see in the next example.

```
import numpy as np
import scipy as sc

from math        import sqrt
from scipy.stats import norm as scnorm
from pylab       import *
from matplotlib  import pyplot as pl

xx = np.array([-0.51, 51.2])
yy = np.array([0.33, 51.6])
means = [xx.mean(), yy.mean()]
stds  = [xx.std()  , yy.std() ]
corr  = 0.75       # correlation
covs  = [[stds[0]**2          , stds[0]*stds[1]*corr],
        [stds[0]*stds[1]*corr,           stds[1]**2]]

m = np.random.multivariate_normal(means, covs, 1000).T
scatter(m[0], m[1])
```

Subsection 2

Brownian simulation of correlated assets

Brownian simulation of correlated assets

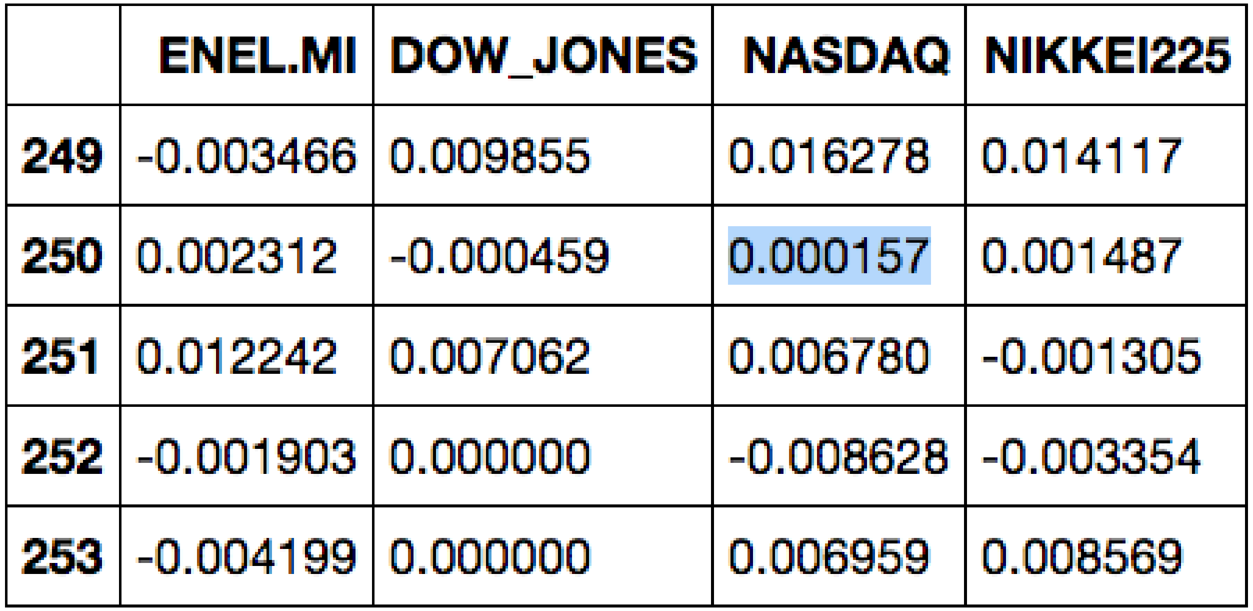
- When using Monte Carlo methods to price options dependent on a basket of underlying assets (multidimensional stochastic simulations), the correlations between assets should be considered.
- Here I will show an example of how this can be simulated using pandas.

```
import pandas as pd
from pandas import Panel, DataFrame

panel_data = pd.read_csv("./data/ts_basket_2.csv", sep=";")
panel_data.tail()
closing = panel_data
```

Brownian simulation of correlated assets

Now we can calculate the log returns:

```
rets = log(closing / closing.shift(1)).dropna()
rets.tail()
```

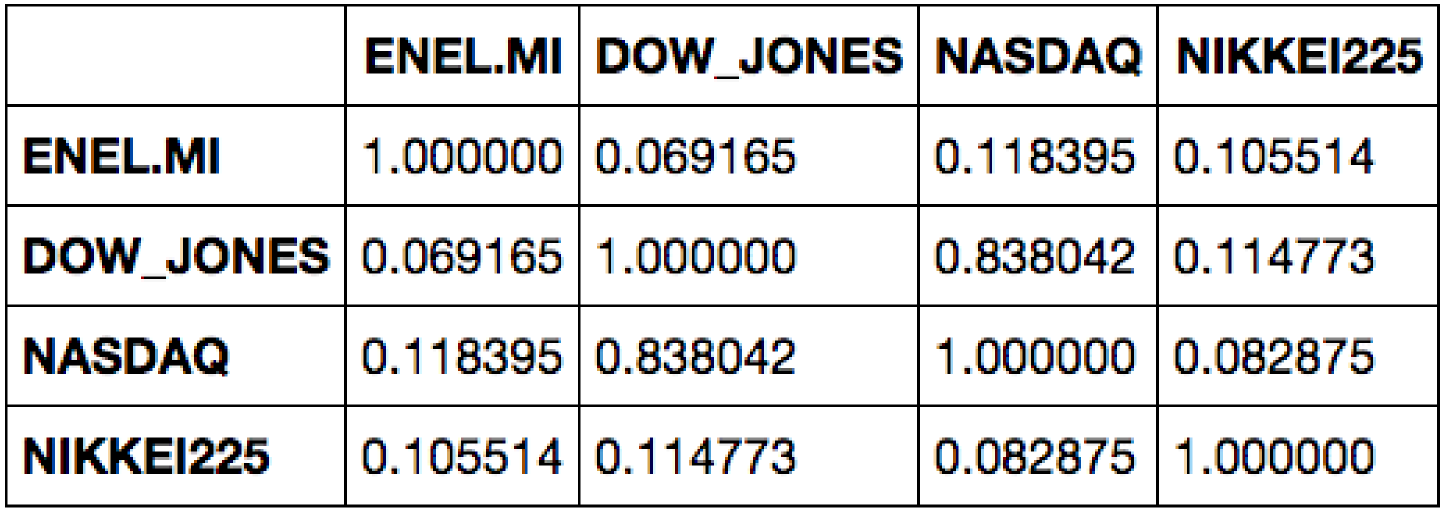


	ENEL.MI	DOW_JONES	NASDAQ	NIKKEI225
249	-0.003466	0.009855	0.016278	0.014117
250	0.002312	-0.000459	0.000157	0.001487
251	0.012242	0.007062	0.006780	-0.001305
252	-0.001903	0.000000	-0.008628	-0.003354
253	-0.004199	0.000000	0.006959	0.008569

Brownian simulation of correlated assets

The correlation matrix has information about the historical correlations between stocks in the group. We work under the assumption that this quantity is conserved, so the generated stocks will need to satisfy this condition:

```
corr_matrix = rets.corr()
corr_matrix
```

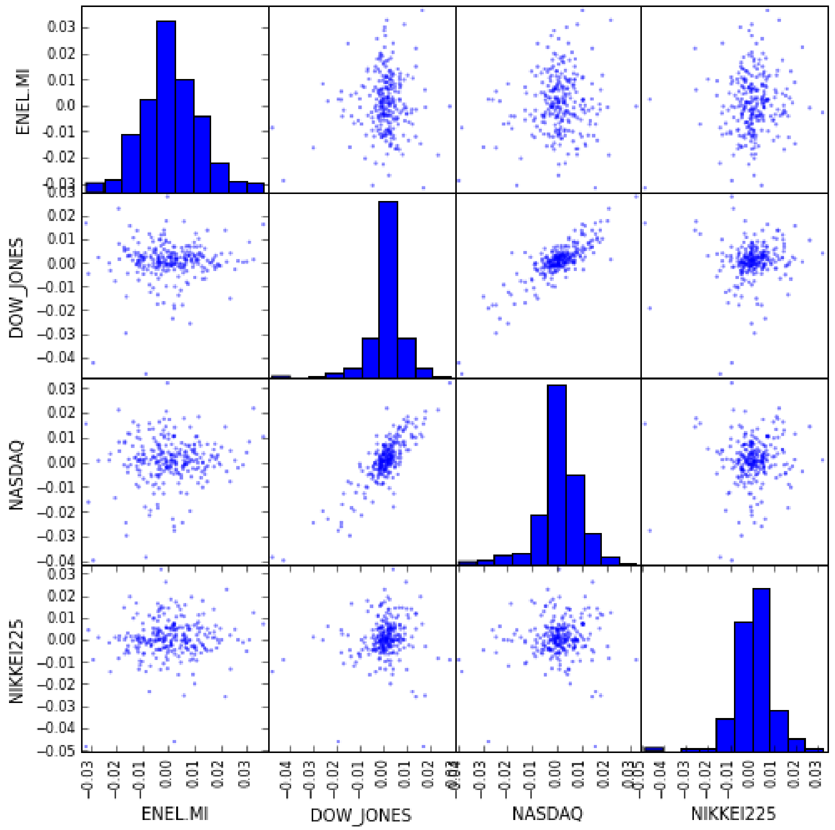


	ENEL.MI	DOW_JONES	NASDAQ	NIKKEI225
ENEL.MI	1.000000	0.069165	0.118395	0.105514
DOW_JONES	0.069165	1.000000	0.838042	0.114773
NASDAQ	0.118395	0.838042	1.000000	0.082875
NIKKEI225	0.105514	0.114773	0.082875	1.000000

Brownian simulation of correlated assets

Pandas has a nice utility to plot the correlations:

```
from pandas.tools.plotting import scatter_matrix
scatter_matrix(rets, figsize=(8,8));
```



Brownian simulation of correlated assets

The simulation procedure for generating random variables will go like this:

- Calculate the Cholesky Decomposition matrix, this step will return an upper triangular matrix L^T.
- Generate random vector $X \sim N(0, 1)$.
- Obtain a correlated random vector $Z = XL^T$.

As we have previously seen the Cholesky decomposition of the correlation matrix is impemented in scipy:

```
from scipy.linalg import cholesky

upper_cholesky = cholesky(corr_matrix, lower=False)
upper_cholesky
```

Brownian simulation of correlated assets

We set up the parameters for the simulation:

```
import numpy as np
from pandas import bdate_range   # business days

n_days   = 21
dates    = bdate_range(start=closing.iloc[-1].name,
                       periods=n_days)
n_assets = 4
n_sims   = 1000
dt       = 1./252
mu       = rets.mean().values
sigma    = rets.std().values*sqrt(252)
# init random number generator for reproducibility
np.random.seed(1234)
```

Brownian simulation of correlated assets

Now we generate the correlated random values X:

```
rand_values =
    np.random.standard_normal(
              size = (n_days * n_sims, n_assets)
              )
corr_values = rand_values.dot(upper_cholesky)*sigma
corr_values
```

Brownian simulation of correlated assets

With everything set up we can start iterating through the time interval. The results for each specific time are saved along the third axis of a pandas Panel.

```
nAsset = 4
symbols = ['ENEL.MI', 'DOW_JONES', 'NASDAQ', 'NIKKEY225']
prices = Panel(items=range(n_sims), minor_axis=symbols,
               major_axis=dates)
prices.iloc[:, 0, :] =
    closing.iloc[1].values.repeat(n_sims).reshape(nAsset, n_sims).T
for i in range(1,n_days):
     prices.iloc[:, i, :] = prices.iloc[:, i-1,:] *
     (exp((mu-0.5*sigma**2)*dt
     + sqrt(dt)*corr_values[i::n_days])).T
# show random path
prices.iloc[123, :, :].head()
```

Notebook

- **GitHub :** polyhedron-gdl;
- **Notebook :** n05_mcs_multi_asset_path;

Subsection 3

Copula Functions

Modelling Dependence with Copulas

- A copula is a function which couples a multivariate distribution function to its marginal distribution functions, generally called marginals or simply margins. Copulas are great tools for modelling and simulating correlated random variables.
- The main appeal of copulas is that by using them you can model the correlation structure and the marginals (i.e. the distribution of each of your random variables) separately.
- This can be an advantage because for some combination of marginals there are no built-in functions to generate the desired multivariate distribution.
- For instance, in R it is easy to generate random samples from a multivariate normal distribution, however it is not so easy to do the same with say a distribution whose margins are Beta, Gamma and Student, respectively.

How Copula works

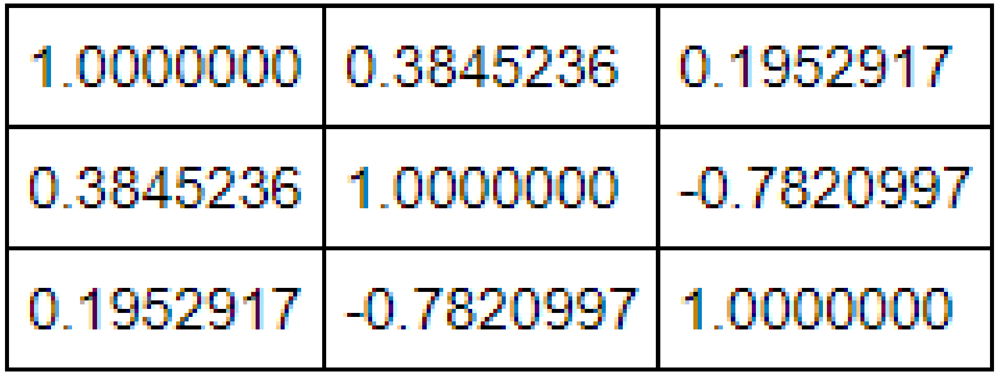
- We generate n samples from a multivariate normal distribution of 3 random variables given the covariance matrix sigma using the MASS package.

```
library(MASS)
set.seed(100)
m <- 3
n <- 5000
sigma <- matrix(c(1, 0.4, 0.2,
                  0.4, 1, -0.8,
                  0.2, -0.8, 1),
                nrow=3)
z <- mvrnorm(n,mu=rep(0, m),Sigma=sigma,empirical=T)
```

How Copula works

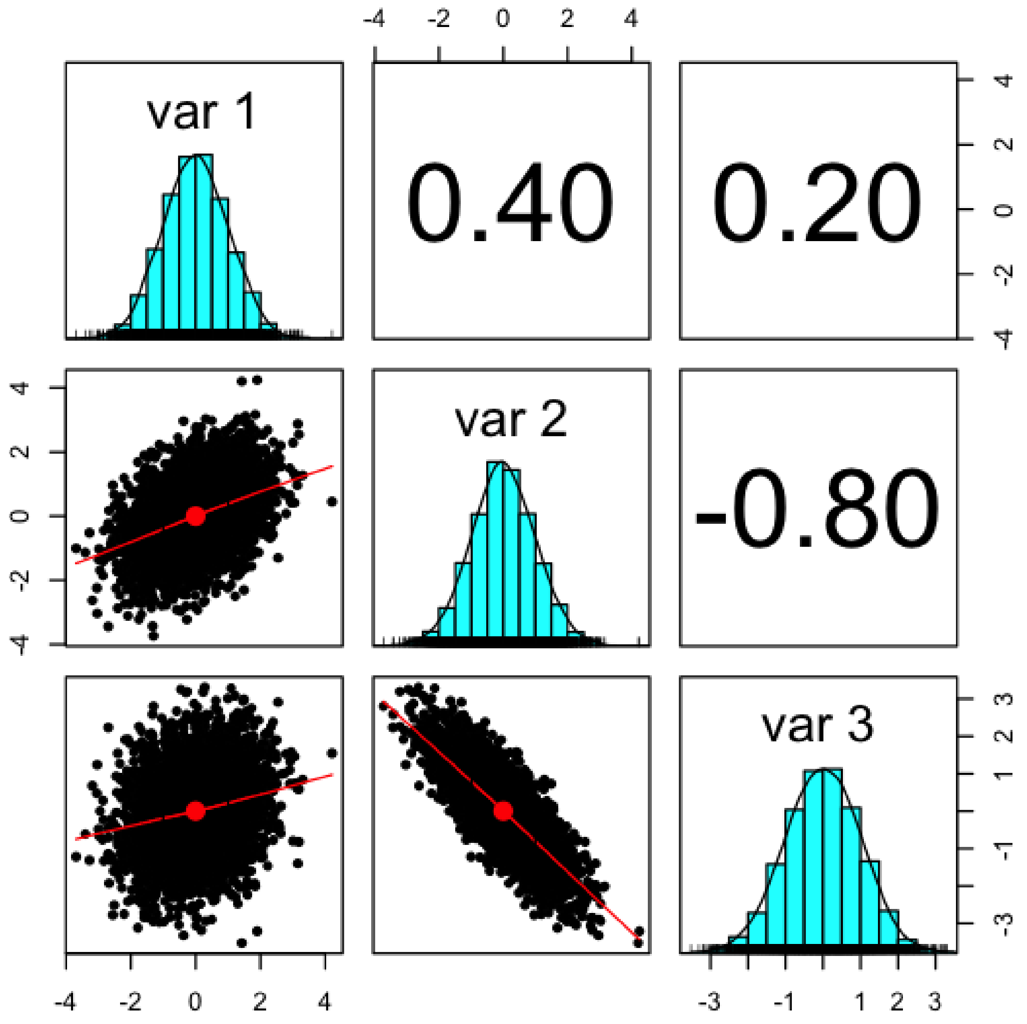
- Now we check the samples correlation using cor() and a pairplot.
- We are going to set method='spearman' in order to use Spearman's Rho instead of the 'pearson' method in the cor() function (this is not strictly necessary in our example since we are going to use an elliptical copula, if you use non-elliptical copulas you may want to use either Spearman's Rho or Kendall's Tau).

```
library(psych)
cor(z,method='spearman')
pairs.panels(z)
```

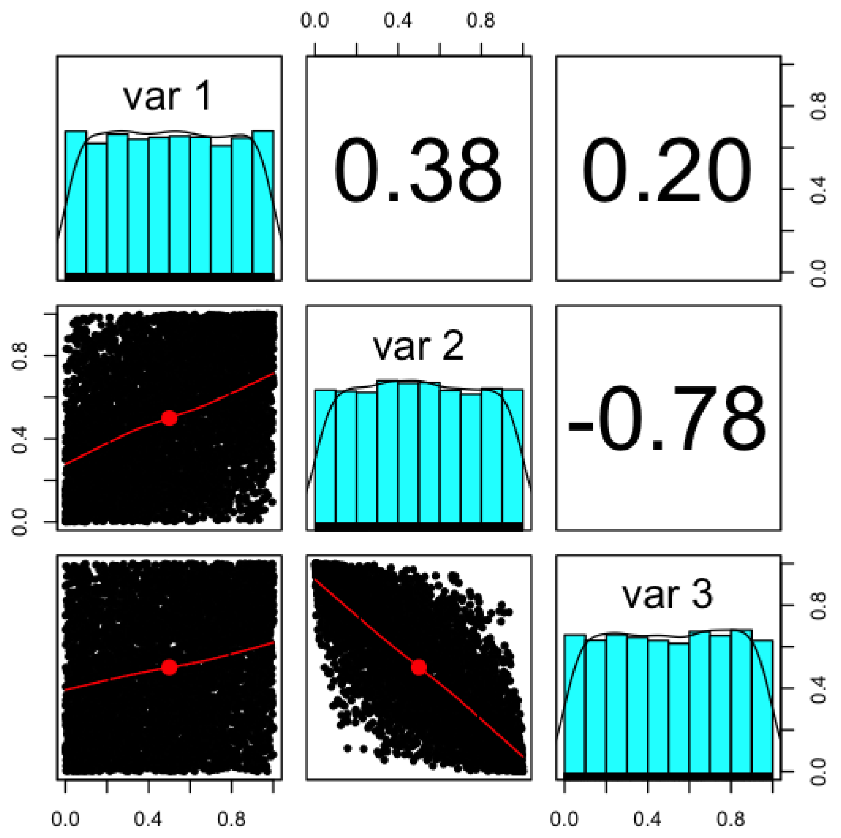


1.0000000	0.3845236	0.1952917
0.3845236	1.0000000	-0.7820997
0.1952917	-0.7820997	1.0000000

How Copula works



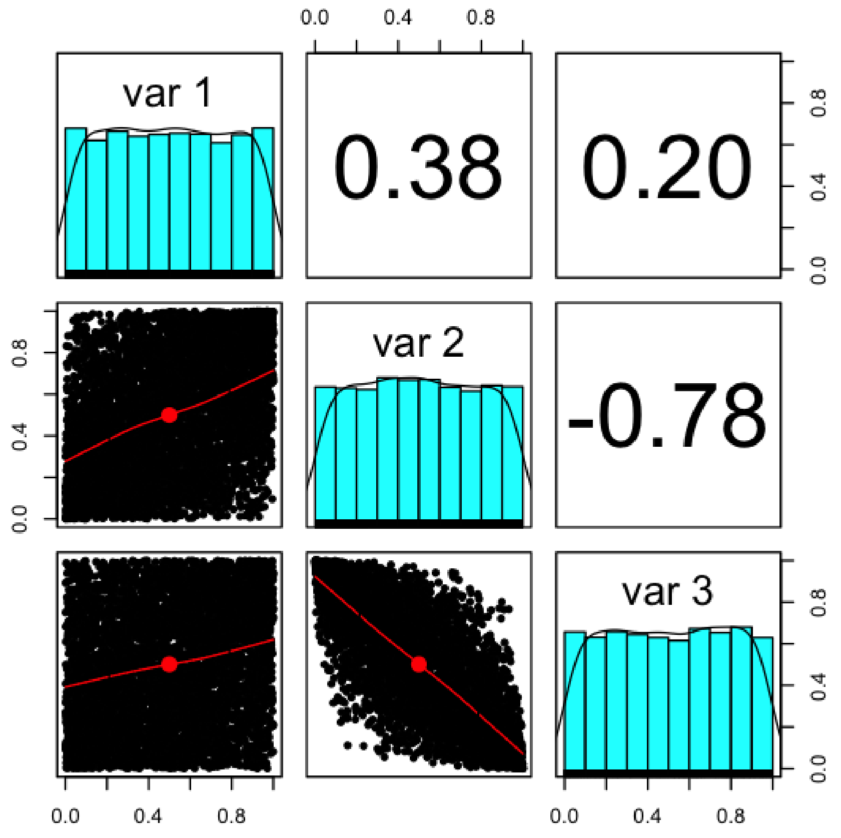
How Copula works



- And now comes the magic trick: recall that if X is a random variable with distribution F then $F(X)$ is uniformly distributed in the interval $[0, 1]$.
- In our toy example we already know the distribution F for each of the three random variables so this part is quick and straightforward.

```
u <- pnorm(z)
pairs.panels(u)
```

How Copula works



- Note that each distribution is uniform in the [0,1] interval.
- Note also that the correlation is the same, in fact, the transformation we applied did not change the correlation structure between the random variables.
- Basically we are left only with what is often referred as the **dependence structure**.

How Copula works

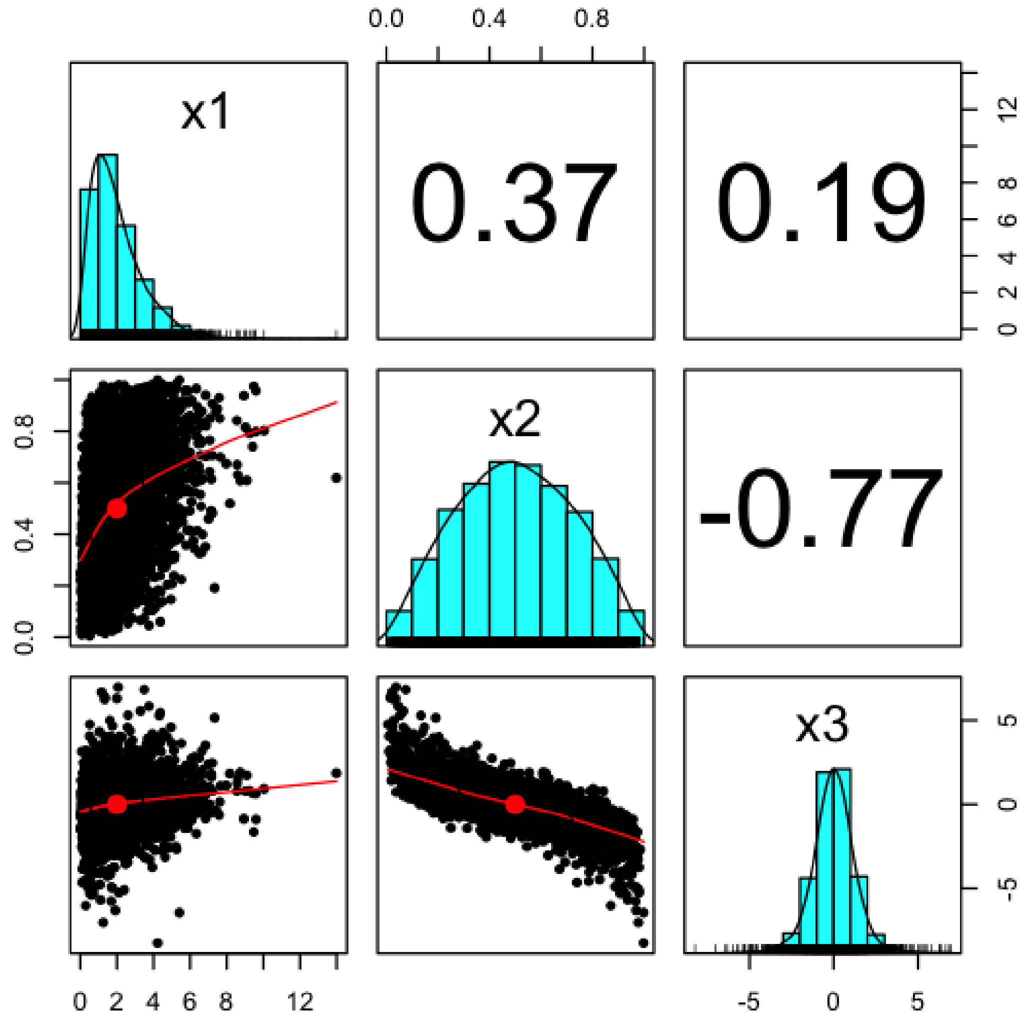
- Now, as a last step, we only need to select the marginals and apply them to u.
- I chose the marginals to be Gamma, Beta and Student distributed with the parameters specified below.

```
x1 <- qgamma(u[,1],shape=2,scale=1)
x2 <- qbeta(u[,2],2,2)
x3 <- qt(u[,3],df=5)
```

- What is worth noticing is that by starting from a multivariate normal sample we have build a sample with the desired and fixed dependence structure and, basically, arbitrary marginals.

```
df <- cbind(x1,x2,x3)
pairs.panels(df)
cor(df,meth='spearman')
```

How Copula works

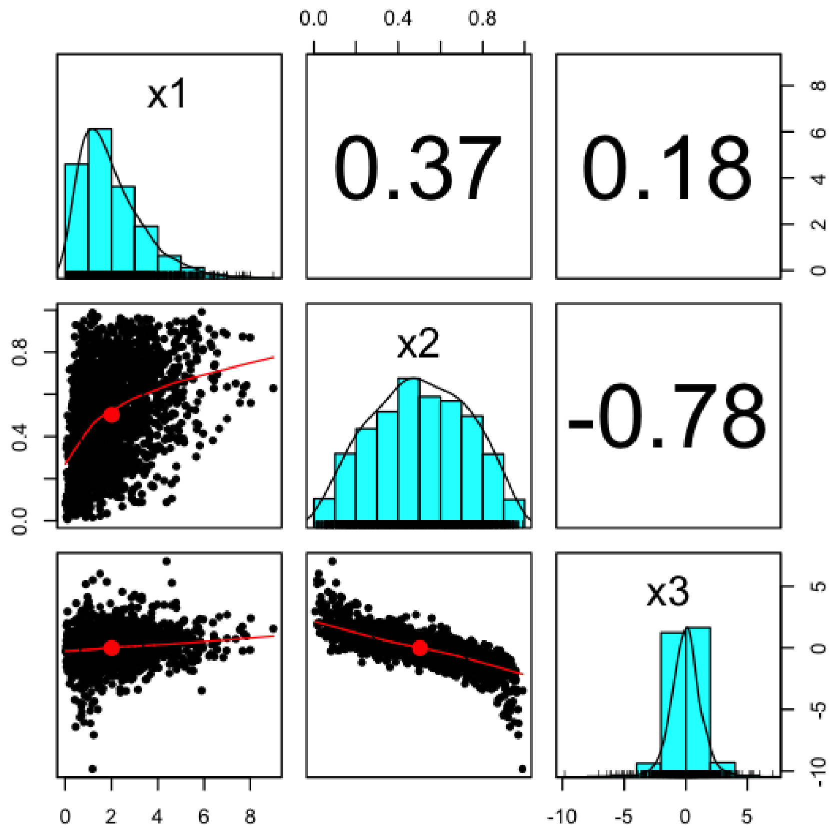


How Copula works

- The whole process performed above can be done more efficiently, concisely and with some additional care entirely by the copula package. Let's replicate the process above using *copulas*.

```
library(copula)
set.seed(100)
myCop <- normalCopula(param=c(0.4,0.2,-0.8), dim = 3,
                                          dispstr = "un")
myMvd <- mvdc(copula=myCop,
                              margins=c("gamma", "beta", "t"),
                   paramMargins=list(
                            list(shape=2, scale=1),
                            list(shape1=2, shape2=2),
                            list(df=5))
                            )
```

How Copula works



- Now that we have specified the dependence structure through the copula (a normal copula) and set the marginals, the mvdc() function generates the desired distribution.
- Then we can generate random samples using the rmvdc() function.

```
Z2 <− rMvdc(2000, myMvd)
colnames(Z2) <− c("x1",
                  "x2",
                  "x3")
pairs.panels(Z2)
```

How Copula works: Gaussian Copula

- The copula of the *n-variate* normal distribution with linear correlation matrix ρ is

$$C_\rho^{Ga}(u_1, \ldots, u_2) = \mathbf{\Phi}_\rho^d(\phi^{-1}(u_1), \ldots, \phi^{-1}(u_d))$$

  where $\mathbf{\Phi}_\rho^d$ denotes the joint distribution function of the n-variate standard normal distribution function with linear correlation matrix ρ, and ϕ^{-1} denotes the inverse of the distribution function of the univariate standard normal distribution.
- Copulas of the above form are called Gaussian copulas.
- The Gaussian copula generates the joint normal standard distribution function iff the margins are standard normals. For any other marginal choice, the Gaussian copula does not give a standard jointly normal vector.

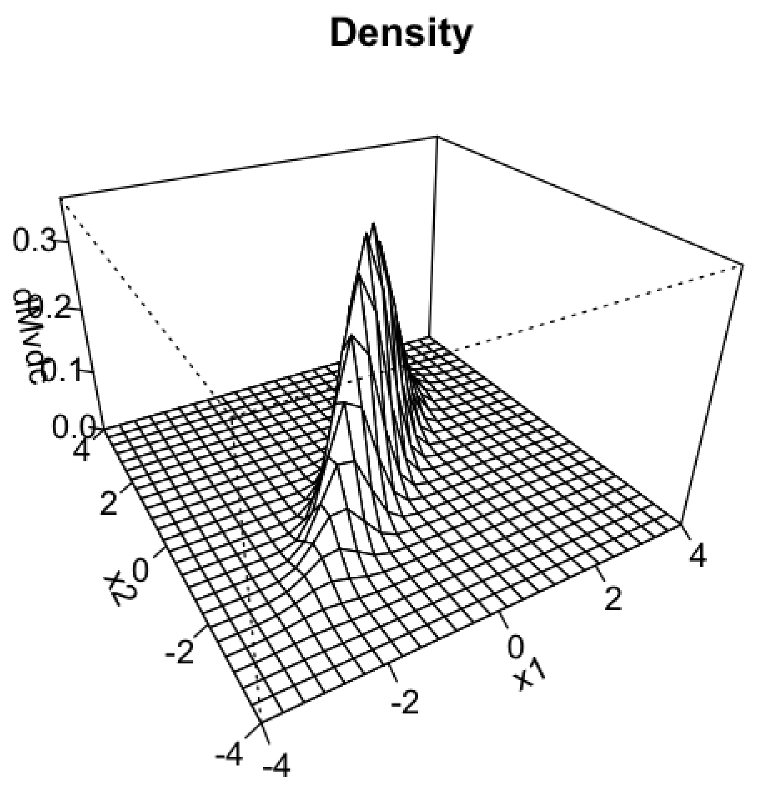
How Copula works

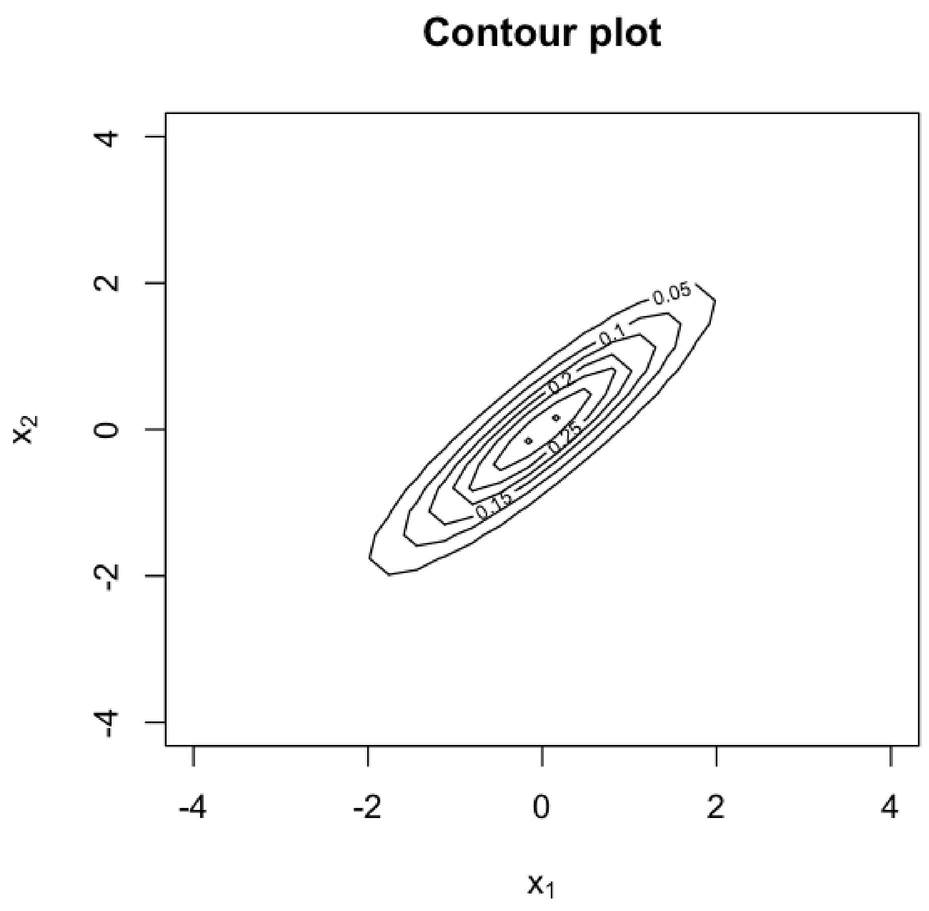
- In order to have a visual representation of the phenomenon and more generally of the effect of "coupling" the same copula with different marginals, let us consider the join density function in the following example in which we analize the coupling of the Gaussian copula with standard gaussian margin and with Student'2 t 3-d.o.f.
- As expected, both in the positive and in the negative correlation cases, the same copula, together with different marginals, present a different joint behavior, here synthesized by the density.
- In the specific case, *the effect of marginal Student distribution is that of increasing the tail probabilities* (although in a symmetric way).

How Copula works

```
library(copula)
rho <- 0.9
# select the copula
cp <- normalCopula(param = rho, dim = 2)
# Density and level curves of the distribution obtained
# coulpling the Gaussian copula with standard
# normal marginals
multivariate_dist <- mvdc(copula      = cp,
                          margins     = c("norm", "norm"),
                          paramMargins = list(list(mean=0,sd=1),
                                              list(mean=0,sd=1)))
par(mfrow=c(1, 2))
persp  (multivariate_dist, dMvdc, xlim = c(-4,4), ylim=c(-4,4),
        main = "Density")
contour(multivariate_dist, dMvdc, xlim = c(-4,4), ylim=c(-4,4),
        main = "Contour plot")
```

How Copula works

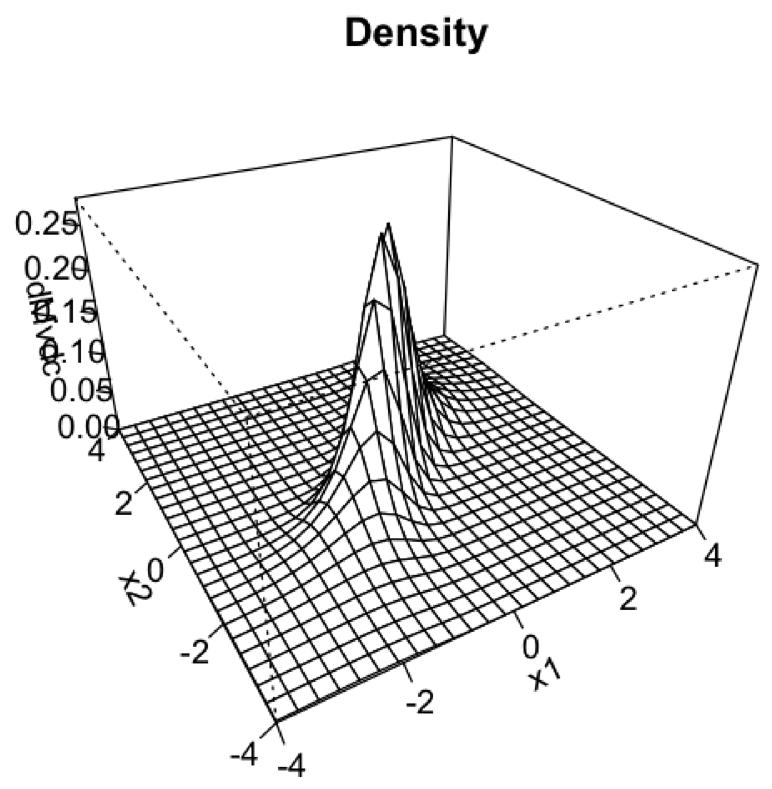


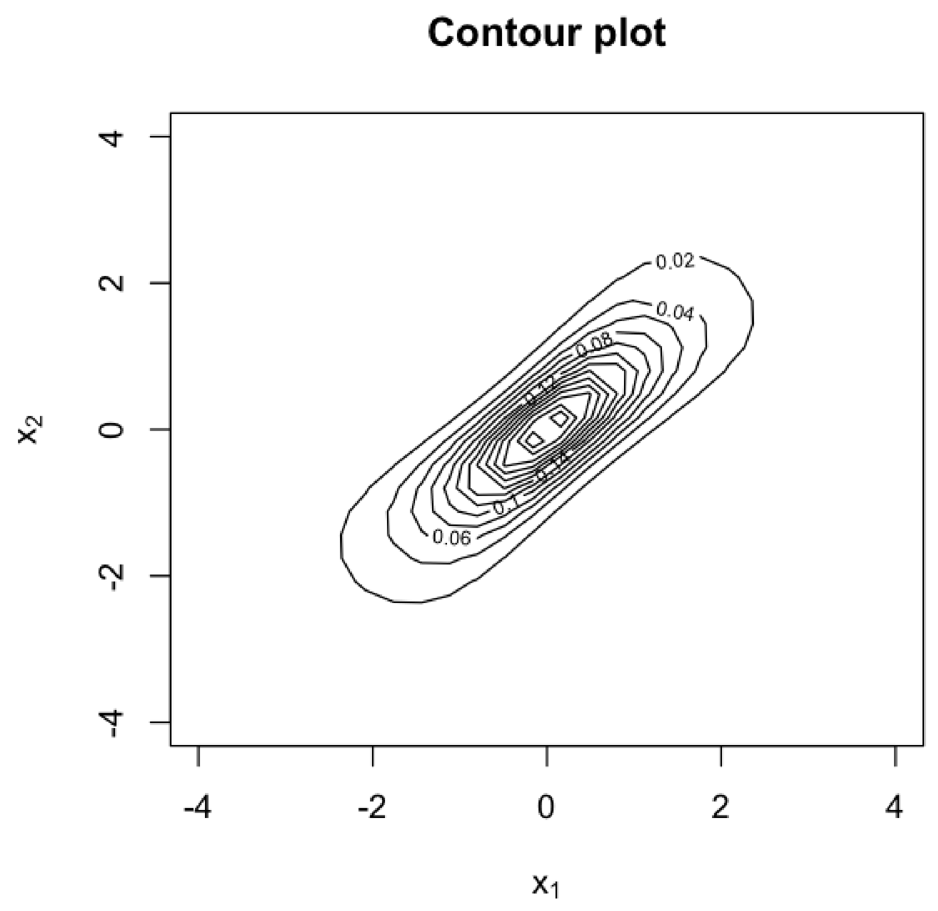


How Copula works

```
# Density and level curves of the distribution obtained
# coulpling the Gaussian copula with 3-d.o.f. Student
multivariate_dist <- mvdc(copula = cp,
                          margins = c("t", "t"),
                          paramMargins = list(list(df = 2),
                                              list(df = 2)) )
par(mfrow=c(1, 2))
persp  (multivariate_dist , dMvdc, xlim = c(-4, 4), ylim=c(-4, 4),
        main = "Density")
contour(multivariate_dist , dMvdc, xlim = c(-4, 4), ylim=c(-4, 4),
        main = "Contour plot")
```

How Copula works





How Copula works: t copula

- Similarly to the case of a normal distribution we can consider the distribution t_ν; the corresponding copula with correlation ρ is:

$$C^t_{\nu,\rho}(u_1, \ldots, u_d) = \Theta^d_{\nu,\rho}(t_\nu^{-1}(u_1), \ldots, t_\nu^{-1}(u_d)) \tag{1}$$

  where $\Theta^d_{\nu,\rho}$ is the t multivariate distribution with dimension d, ν degree of freedom and linear correlation ρ while t_ν^{-1} is the inverse univariate standard t-distribution t_ν.

- Also for this functional you have a simple simulation algorithm based on the following result: If X has the stochastic representation

$$X = \frac{\sqrt{\nu}}{\sqrt{Z}} Y \quad \text{with} \quad Y \sim \mathcal{N}_d(0, \rho), Z \sim \chi^2_\nu$$

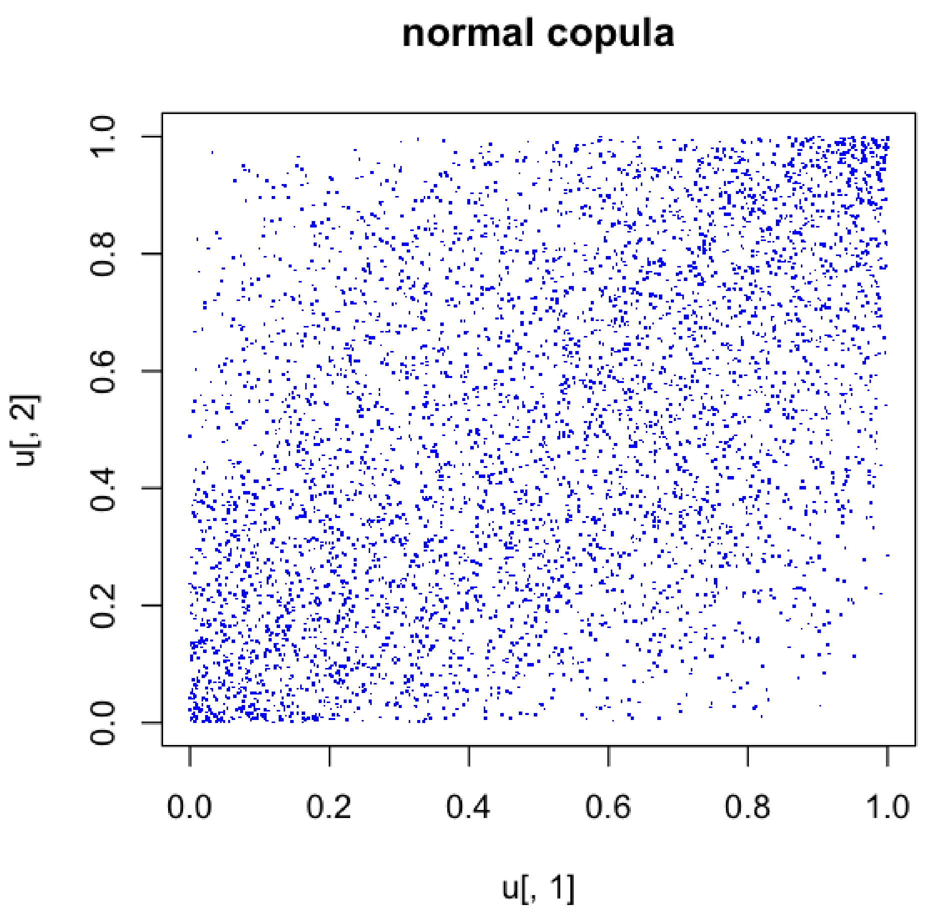
  then X has an $n-variate$ t_ν-distribution with mean μ (for $\nu > 1$) and covariance matrix $\nu\Sigma/(\nu - 2)$ (for $\nu > 2$). If $\nu \leq 2$ then $Cov(X)$ is not defined. In this case we just interpret Σ as being the shape parameter of the distribution of X.

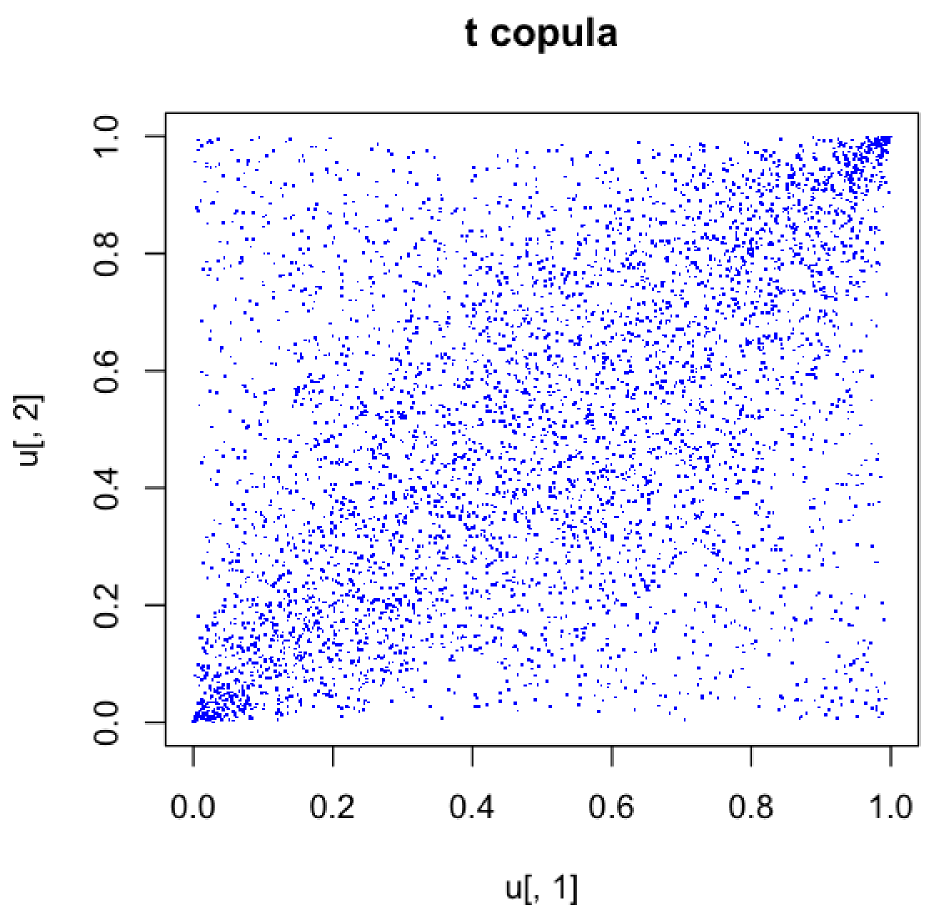
How Copula works

Now generate some samples from gaussian and t copulas.

```
rho <- 0.5
u <- rCopula(5000,normalCopula(dim=2,rho))
plot(u[,1],u[,2],pch='.',col='blue',
        main="normal copula")

u <- rCopula(5000,tCopula(dim=2,rho,df=2))
plot(u[,1],u[,2],pch='.',col='blue',
        main="t copula")
```

How Copula works





Subsection 4

A Practical Example with Copula Functions

Simulation of a Two Stock Portfolio Return

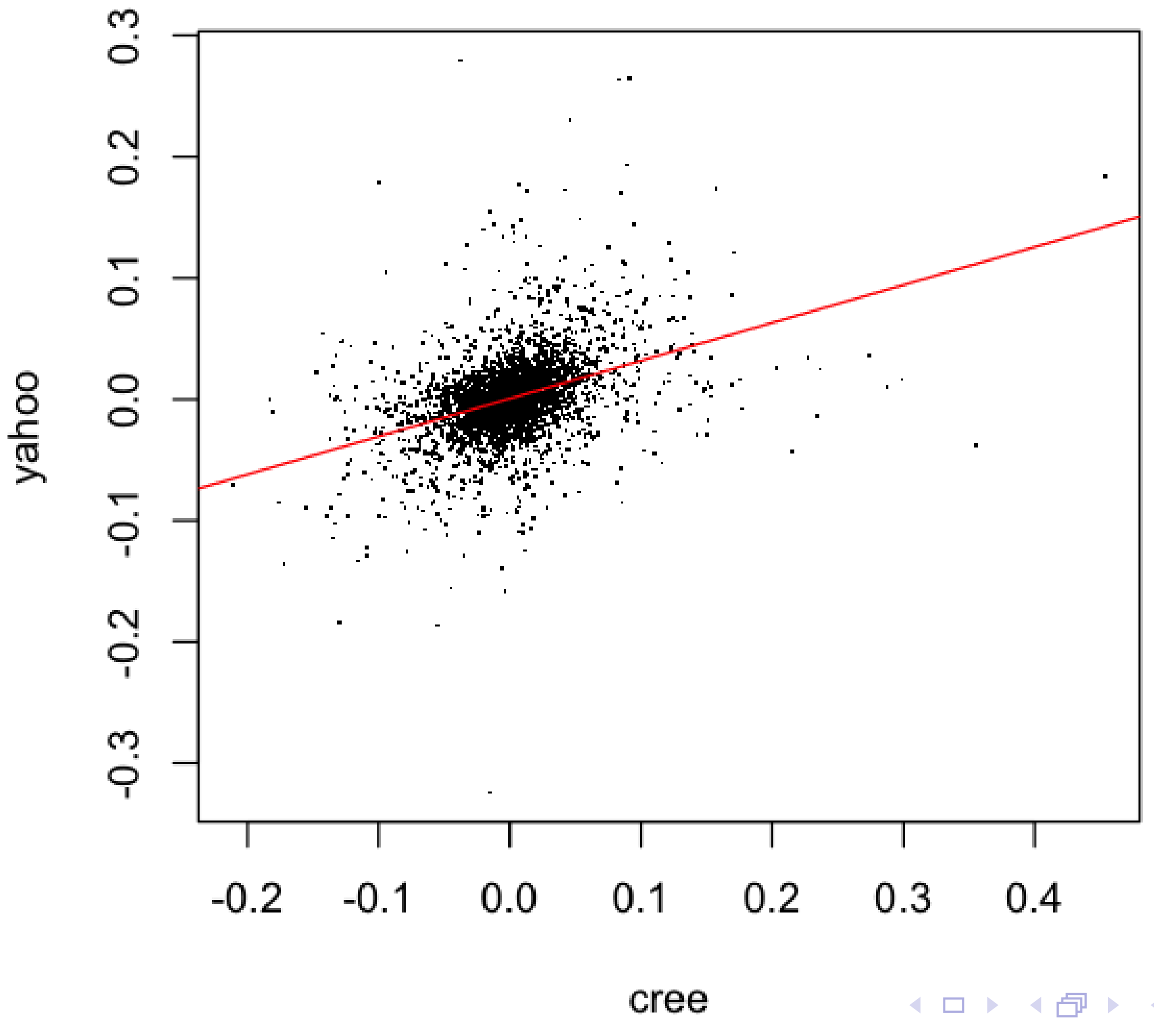
- Now a real world example although a very simple one.
- We are going to fit a copula to the returns of two stocks and try to simulate the returns using the copula.
- Let's load the returns in R

```
cree  <- read.csv('cree_r.csv',  header=F)$V2
yahoo <- read.csv('yahoo_r.csv',header=F)$V2
```

- Before going straight to the copula fitting process, let's check the correlation between the two stock returns and plot the regression line:

```
plot(cree,yahoo,pch='.')
abline(lm(yahoo~cree),col='red',lwd=1)
cor(cree,yahoo,method='spearman')
```

How Copula works



Simulation of a Two Stock Portfolio Return

- We can see a very mild positive correlation, the returns look mainly like a random blob on the plot.
- When applying these models to real data one should carefully think at what could suit the data better.
- For instance, many copulas are better suited for modelling asymetrical correlation other emphasize tail correlation a lot, and so on.
- My guess for stock returns is that a t-copula should be fine, however a guess is certainly not enough.
- Fortunately, the VineCopula package offers a great function which tells us what copula we should use.
- Essentially the VineCopula library allows us to perform copula selection using BIC and AIC through the function BiCopSelect()

Simulation of a Two Stock Portfolio Return

```
library(VineCopula)
u <- pobs(as.matrix(cbind(cree,yahoo)))[,1]
v <- pobs(as.matrix(cbind(cree,yahoo)))[,2]
selectedCopula <- BiCopSelect(u,v,familyset=NA)
selectedCopula

Bivariate copula: t (par = 0.44, par2 = 3.84, tau = 0.29)
```

- Note that I fed into the selection algorithm the pseudo observations using the pobs() function. Pseudo observations are the observations in the [0,1] interval.
- The fitting algorithm indeed selected a t-copula (encoded as 2 in the family reference) and estimated the parameters for us.

Simulation of a Two Stock Portfolio Return

- Let's try to fit the suggested model using the copula package and double check the parameters fitting.

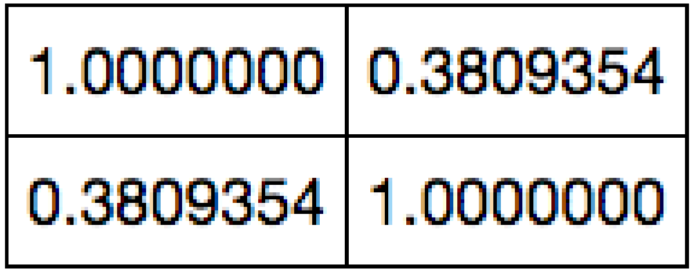
```
t.cop <- tCopula(dim=2)
set.seed(500)
m      <- pobs(as.matrix(cbind(cree,yahoo)))
fit    <- fitCopula(t.cop,m, method='ml')
coef(fit)

rho.1
0.435630004089189
df
3.84452707689695
```

- It is nice to see that the parameters of the fitted copula are the same as those suggested by the BiCopSelect() function.

Simulation of a Two Stock Portfolio Return

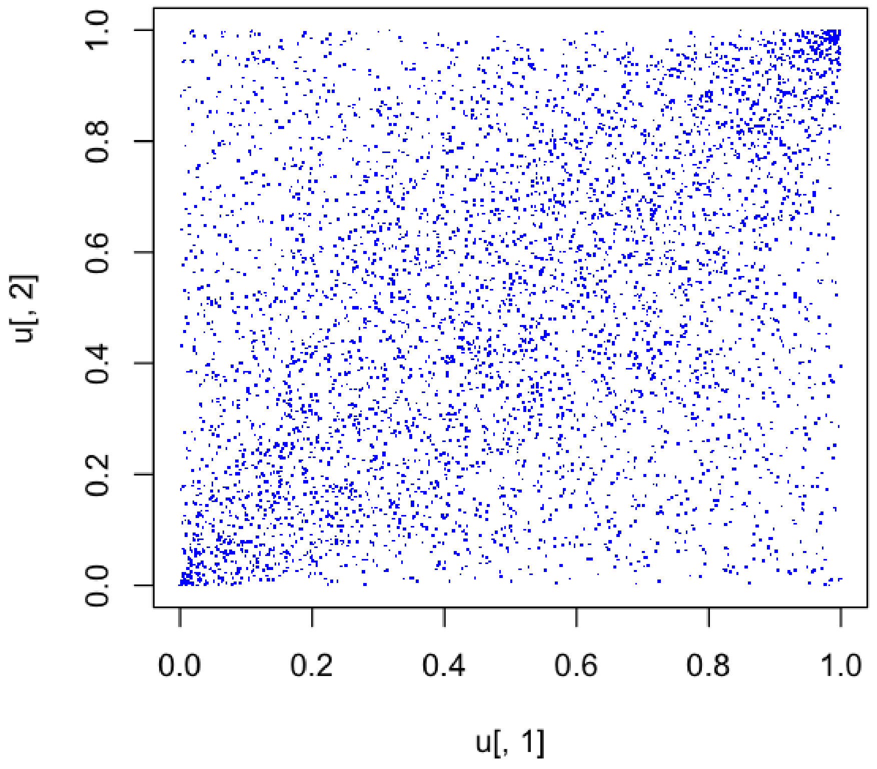
- Now we only need to build the copula and sample from it 5000 random samples.

```
u <- rCopula(5000,tCopula(dim=2,rho,df=df))
plot(u[,1],u[,2],pch='.',col='blue')
cor(u,method='spearman')
```



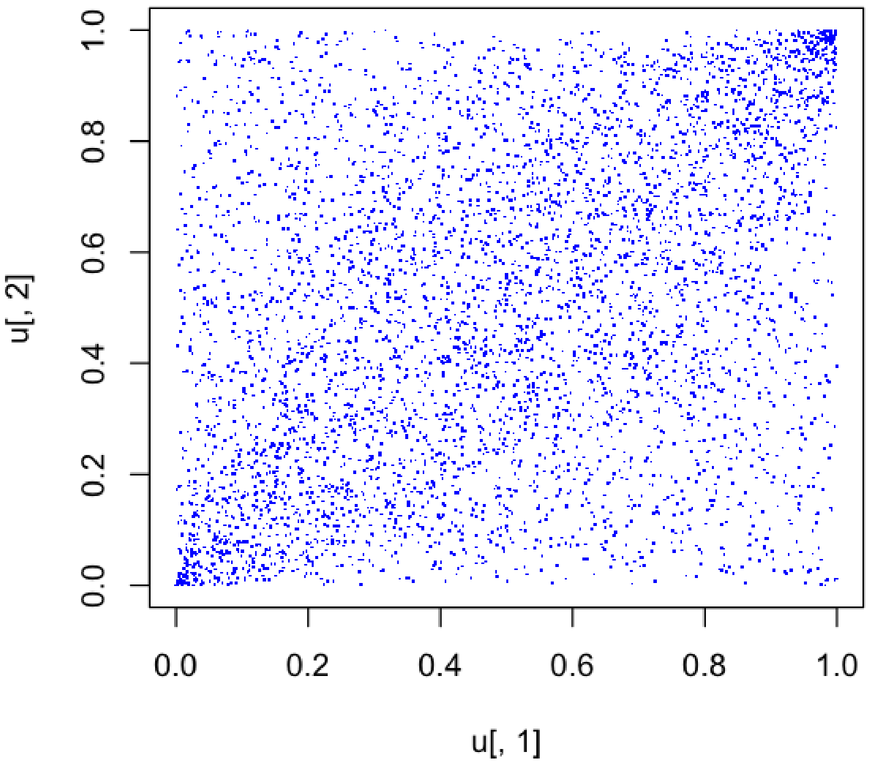
1.0000000	0.3809354
0.3809354	1.0000000

How Copula works



- This is the plot of the samples contained in the vector u.
- The random samples from the copula look a little bit close to the independence case, but that is fine since the correlation between the returns is low.
- Note that the generated samples have the same correlation as the data.

How Copula works



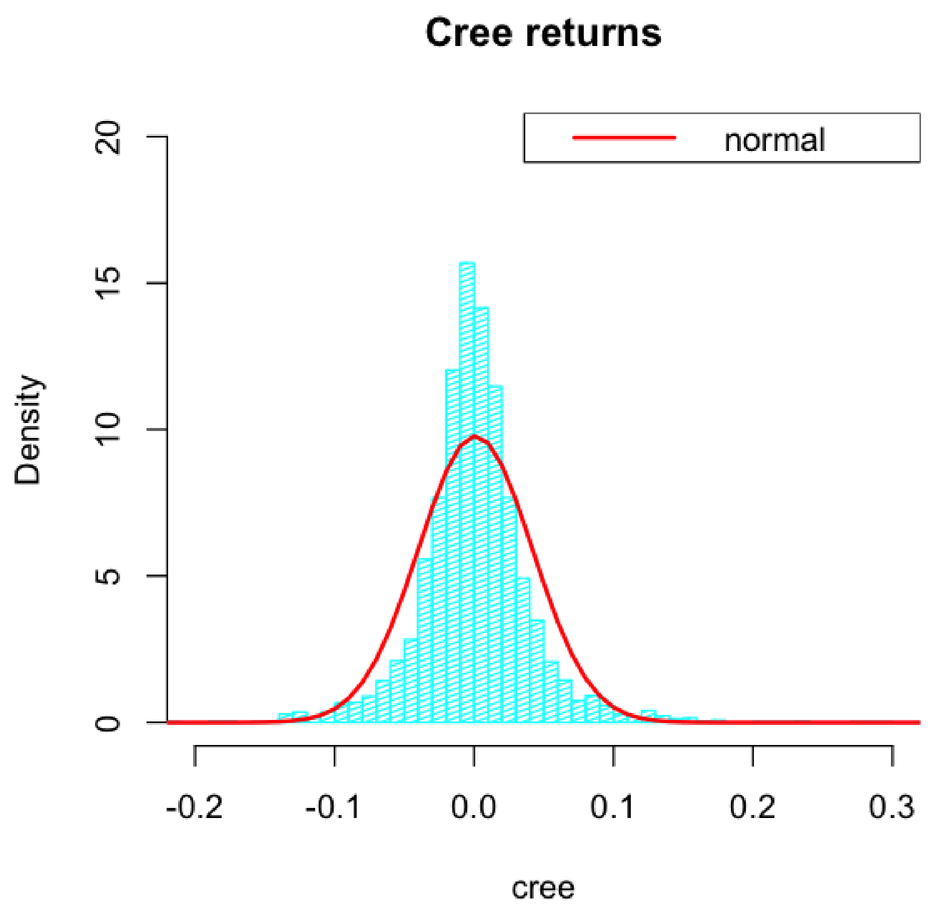
- The t-copula emphasizes extreme results: it is usually good for modelling phenomena where there is high correlation in the extreme values (the tails of the distribution).
- Note also that it is symmetrical, this might be an issue for our application, however we are going to neglect this.

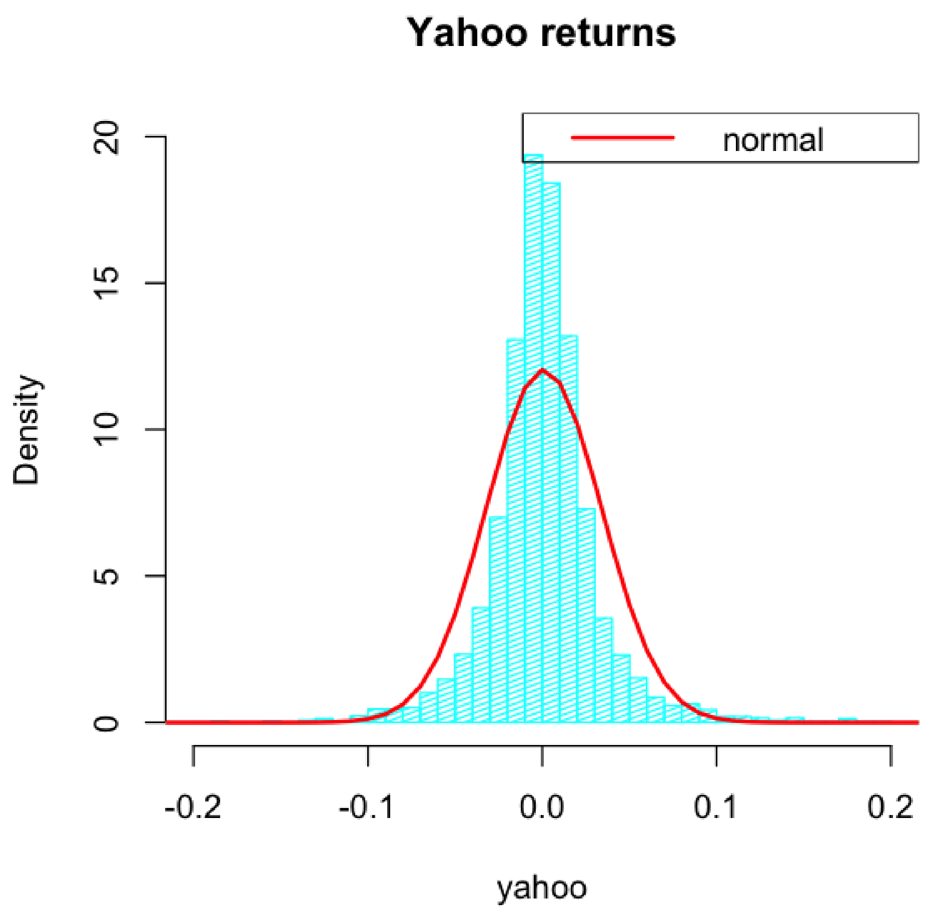
Simulation of a Two Stock Portfolio Return

Now we are facing the hard bit: modelling the marginals. We are going to assume normally distributed returns for simplicity even though it is well known to be a far from sound assumption. We therefore estimate the parameters of the marginals

```
cree_mu  <- mean(cree)
cree_sd  <- sd(cree)
yahoo_mu <- mean(yahoo)
yahoo_sd <- sd(yahoo)

options(repr.plot.width=10, repr.plot.height=5)
par(mfrow=c(1, 2))

hist(cree,breaks=80,main='Cree returns',freq=F,density=30,col='cyan',ylim=c(0,20),xlim=c(-0.2,0.3))
lines(seq(-0.5,0.5,0.01),dnorm(seq(-0.5,0.5,0.01),cree_mu,cree_sd),col='red',lwd=2)
legend('topright',c('normal'),col=c('red'),lwd=2)

hist(yahoo,breaks=80,main='Yahoo returns',density=30,col='cyan',freq=F,ylim=c(0,20),xlim=c(-0.2,0.2))
lines(seq(-0.5,0.5,0.01),dnorm(seq(-0.5,0.5,0.01),yahoo_mu,yahoo_sd),col='red',lwd=2)
legend('topright',c('normal'),col=c('red'),lwd=2)
```

Simulation of a Two Stock Portfolio Return





Simulation of a Two Stock Portfolio Return

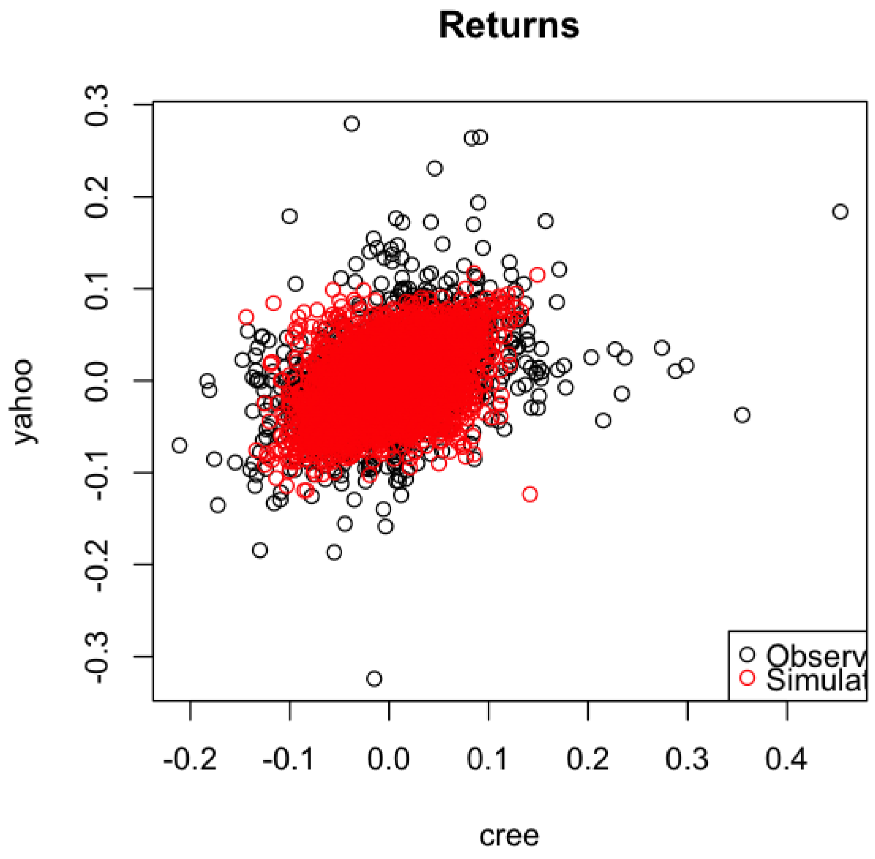
Now we apply the copula in the mvdc() function and then use rmvdc() to get our simulated observations from the generated multivariate distribution.

```
copula_dist <- mvdc(copula=tCopula(rho,dim=2,df=df),
                    margins=c("norm","norm"),
                    paramMargins=list(
                      list(mean=cree_mu,sd=cree_sd),
                      list(mean=yahoo_mu,sd=yahoo_sd)
                      )
                    )
sim <- rMvdc(5000,copula_dist)
```

Simulation of a Two Stock Portfolio Return

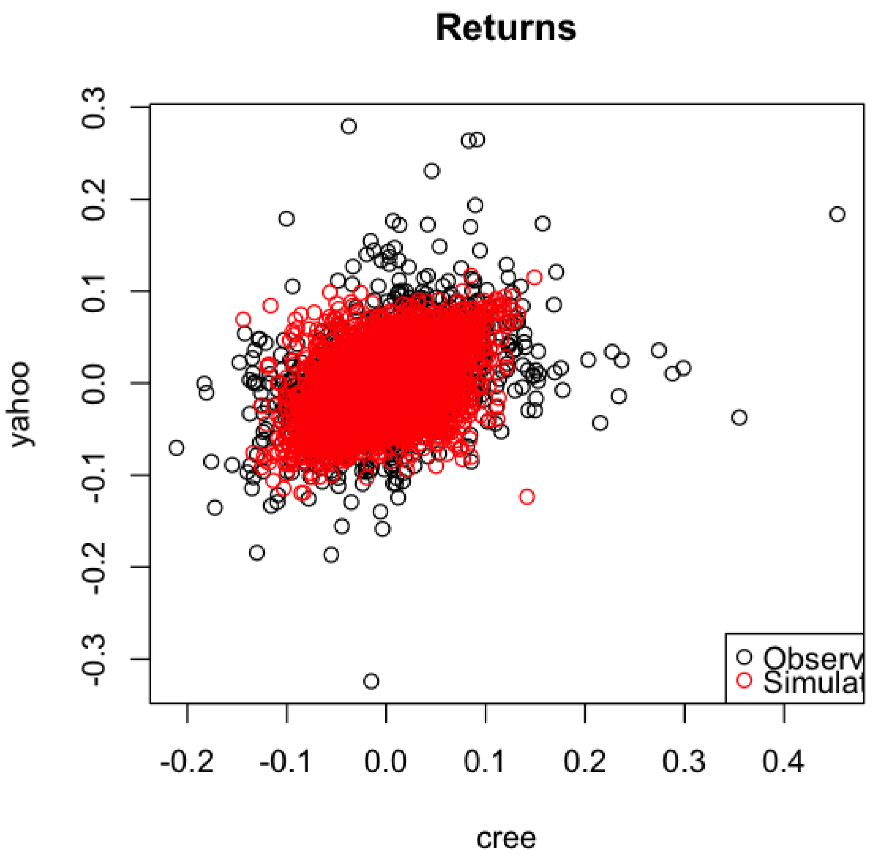
Finally, we compare the simulated results with the original data.

```
plot(cree,
        yahoo,
        main='Returns')
points(sim[,1],
              sim[,2],
              col='red')
legend('bottomright',
    c('Observed',
        'Simulated'),
    col=c('black','red'),
    pch=21)
```



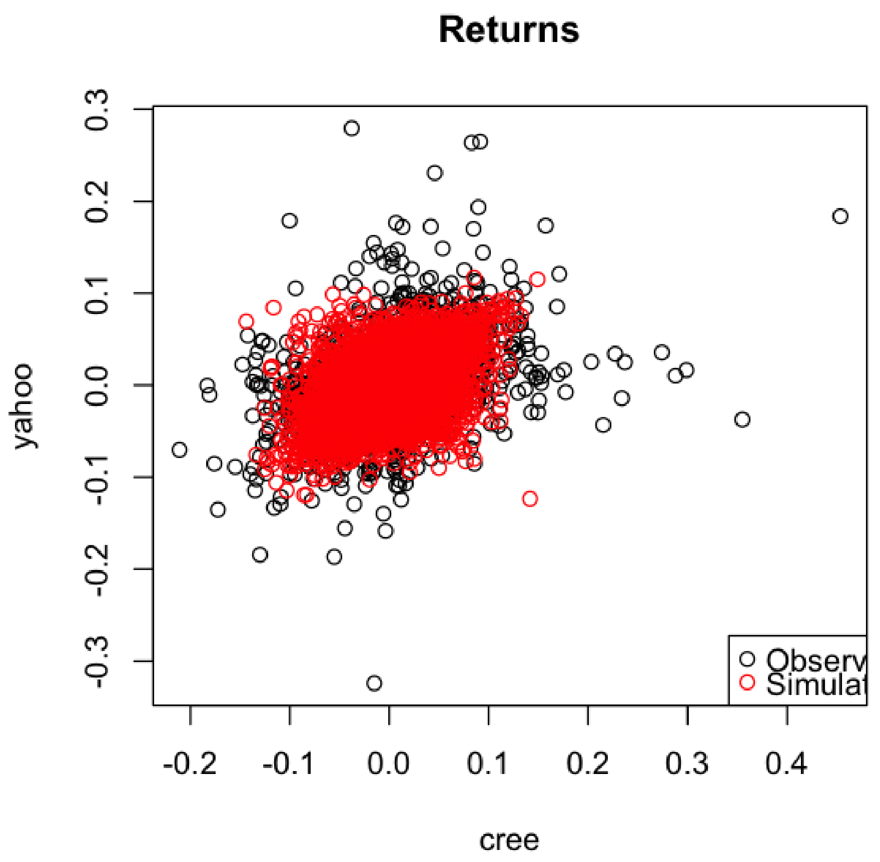
Simulation of a Two Stock Portfolio Return

- This is the final scatterplot of the data under the assumption of normal marginals and t-copula for the dependence structure.
- As you can see, the t-copula leads to results close to the real observations, although there are fewer extreme returns than in the real data and we missed some of the most extreme result.



Simulation of a Two Stock Portfolio Return

- If we were interested in modelling the risk associated with these stocks then this would be a major red flag to be addressed with further model calibration.
- Since the purpose of this example is simply to show how to fit a copula and generate random observations using real data, we are going to stop here.


Returns
yahoo
cree
○ Observ
○ Simulat