3.1.Multivariate Return Modelling & Copulas

This is sourced from STAD70 course practice questions and sample R code taught by professor Sotos. If you have any questions/concerns/comments feel free to email me: cristal.wang111@gmail.com (mailto:cristal.wang111@gmail.com).

1.Multivariate Return Modelling

1.1.Data and Return

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
ticker.symbols= c("KLAC","JNJ","DUK","PNC","AAPL")
N.stocks = length(ticker.symbols)

start.date=as.Date("2003-08-01")  # Prices from Aug 2003 to Aug 2013
end.date=as.Date("2013-08-01")
P=get.hist.quote(instrument = ticker.symbols[1], start = start.date, end=end.date, quote = "AdjC
lose", retclass = "zoo", quiet = T)
names(P)=ticker.symbols[1]

for (i in 2:N.stocks) {
  cat("Downloading ", i, " out of ", N.stocks , "\n")
  x=try(get.hist.quote(instrument = ticker.symbols[i], start = start.date, end=end.date, quote =
"AdjClose", retclass = "zoo", quiet = T))
  if( class(x)!= "try-error"){
    names(x)=ticker.symbols[i]
    P = merge(P, x, all=TRUE)
  }
}
```

```
## Downloading  2  out of  5
## Downloading  3  out of  5
## Downloading  4  out of  5
## Downloading  5  out of  5
```

```
R=diff(log(P)) # Calculate log returns
head(R,3) #Input: data R with several columns (e.g., each column has returns of an asset)
```

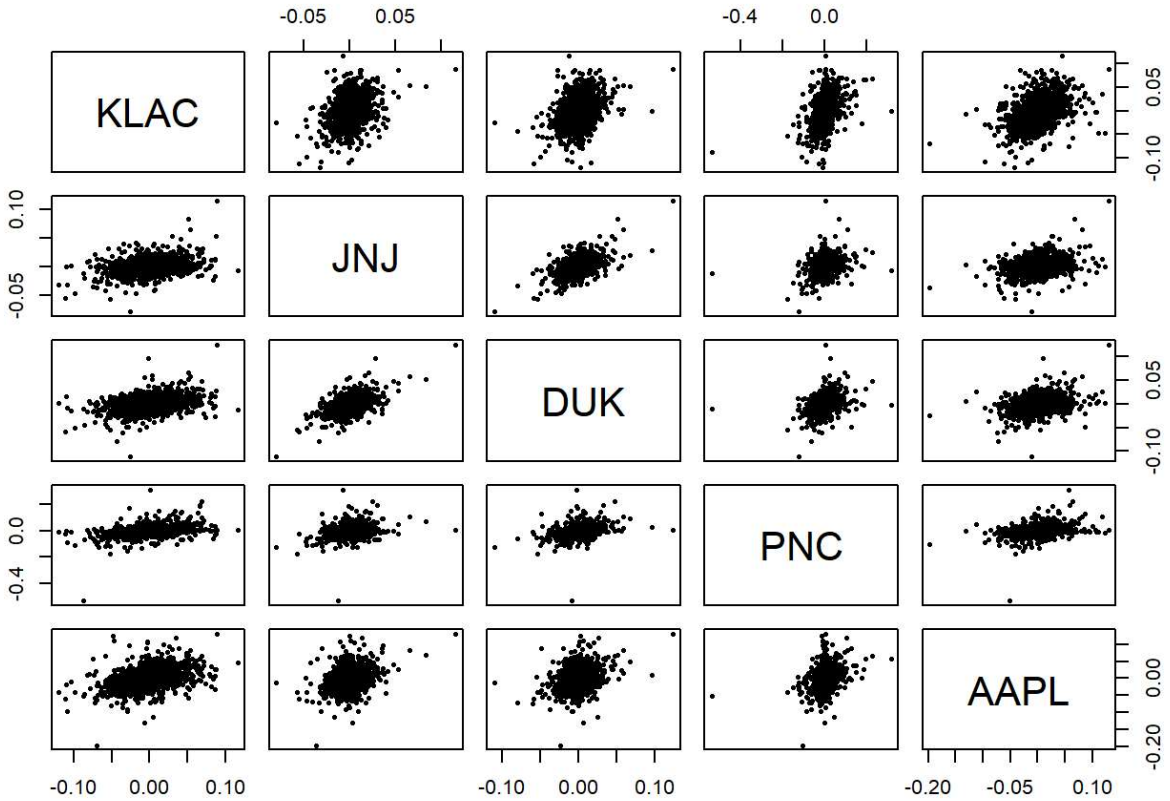
```
##                   KLAC          JNJ          DUK          PNC         AAPL
## 2003-08-04  0.006871441  0.003364969 -0.012709883  0.002059656  0.02289020
## 2003-08-05 -0.038589189 -0.016337906 -0.013461840 -0.025634962 -0.03991786
## 2003-08-06 -0.005750279  0.008600427  0.004116411  0.004423652 -0.03749507
```

1.2.Sample Covariance Estimation

Multivariate Normal Estimation

1.2.1.Scatterplot Matrix of Return

```
pairs(R, pch=16, cex=.5)
```



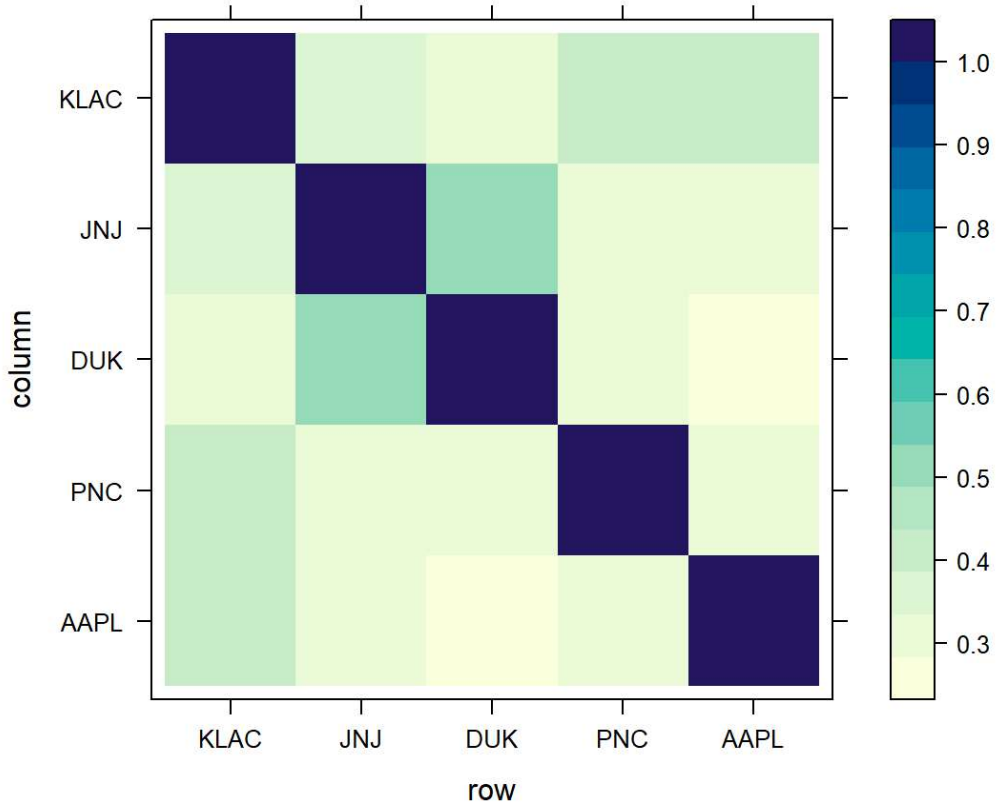
1.2.2.Sample Covariance and correlation matrix

```
COV=cov(R) # sample covariance matrix
COR=cov2cor(COV) # sample correlation matrix (from covariance)
```

1.2.3.Levelplot of correlation matrix

darker => higher correlation

```
library(lattice)
levelplot( COR[,ncol(R):1]  ) # [,ncol(R):1]  means reverse, can remove
```



1.3.Robust estimation

Robust means it is not significantly influenced by outliers.

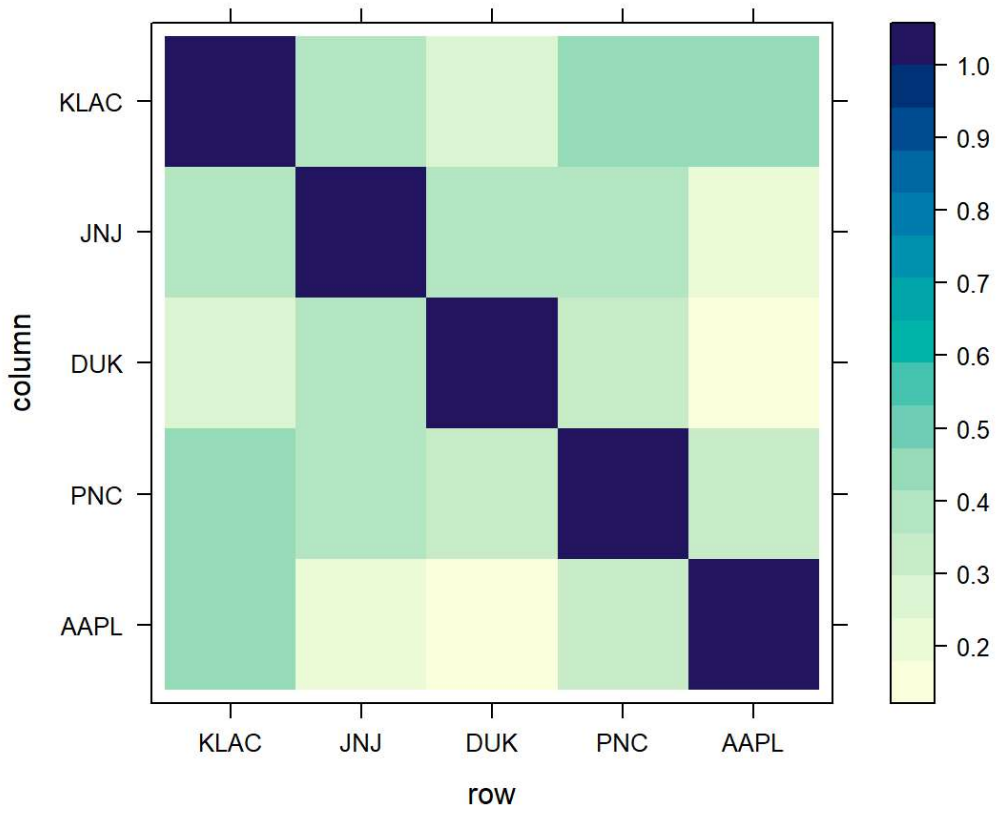
```
library(MASS)
```

1.3.1.Robust estimator of covariance matrix

```
out=cov.rob(R)
COV.rob=out$cov
```

1.3.2.level plot of robust correlation matrix

```
COR.rob=cov2cor(COV.rob)
levelplot( COR.rob[,ncol(R):1]  )
```



1.4.Multivariate Student's t estimation and simulation

- Never remove outliers in finance ⇒ model heavy tails using *multivariate* t distribution
  - *Multivariate* Normal scale mixture

$$R \sim t_v(\mu, \Lambda); \quad R = \mu + Z\sqrt{\nu/W}, \quad \text{where} \begin{cases} W \sim \chi^2(df = v) \\ Z \sim N(0, \Lambda), \end{cases} \quad \text{where } E[R] = \mu,\ Cov[R] = \Lambda \cdot \frac{v}{v-2}$$

- Marginals are t-distributed with same df => all assets returns have same tail index
- Model exhibits tail dependence

```
library(mnormt)
```

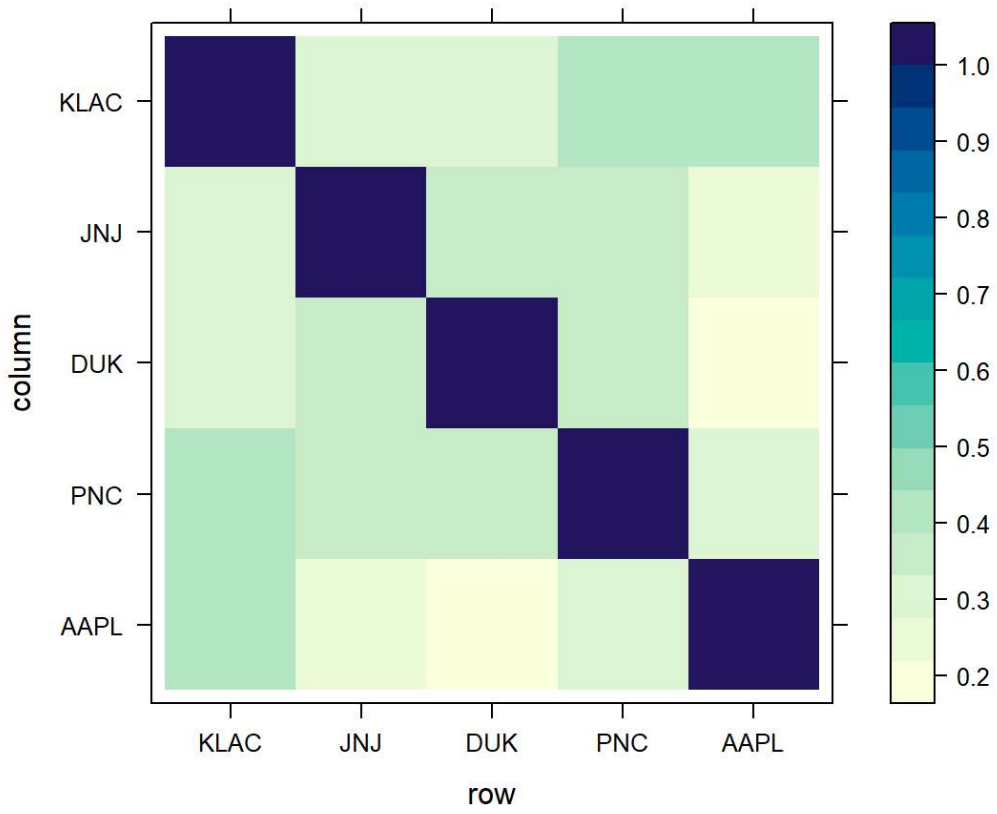
1.4.1.Multivariate t estimation (fit t, find df by MLE)

- `cov.trob` : It stands for **covariance estimation using a t-distribution** (hence, "t-robust"). The method assumes your data comes from a multivariate t-distribution, not a normal distribution. Useful when your data has heavy tails or outliers.
- `fit = cov.trob(R, nu = df.opt)`
  - `fit$cov` : the estimated robust covariance matrix
  - `fit$center` : the estimated mean vector

```
library(MASS) # needed for cov.trob
library(mnormt) # needed for dmt

## range of degree of freedom
df = seq(1.5,8,by=.2)
n = length(df)

## for each df in range,fit robust covariance matrix, calculate the corresponding log-likelihoo
d, store in loglik
loglik = rep(0,n)
for(i in 1:n){
  fit = cov.trob(R,nu=df[i])
  loglik[i] = sum(log(dmt(R,mean=fit$center,S=fit$cov,df=df[i])))
}
## take the df(nu) which maximize the log-likelihood
df.opt=df[which.max(loglik)]
fit = cov.trob(R,nu=df.opt)

fit$cov * (df.opt/(df.opt-2)) # t-distribution's covariance matrix df/(df-2)
```

```
##              KLAC          JNJ          DUK          PNC         AAPL
## KLAC 6.331922e-04 8.132398e-05 9.112617e-05 2.334500e-04 2.782847e-04
## JNJ  8.132398e-05 1.017487e-04 5.094939e-05 7.814252e-05 5.868427e-05
## DUK  9.112617e-05 5.094939e-05 1.715105e-04 9.745096e-05 7.317595e-05
## PNC  2.334500e-04 7.814252e-05 9.745096e-05 4.960481e-04 1.694017e-04
## AAPL 2.782847e-04 5.868427e-05 7.317595e-05 1.694017e-04 6.556446e-04
```

```
levelplot( cov2cor(fit$cov)[,ncol(R):1] )# t-distribution's correlation matrix
```

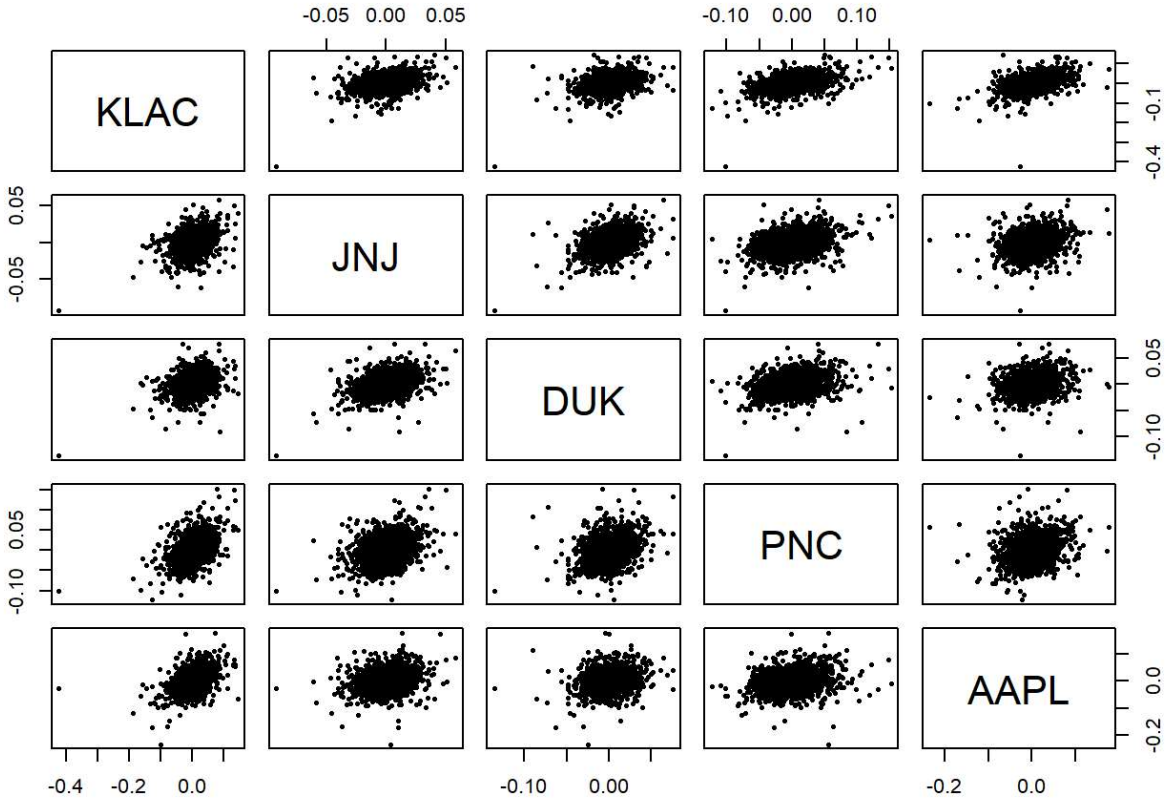


1.4.2.Simulate multivariate t

Use

$$\mathbf{t} = \boldsymbol{\mu} + \mathbf{Z}\sqrt{\frac{\nu}{W}}$$

```
Sim_Number = 5000
Z=rmnorm(Sim_Number,rep(0,ncol(R)),fit$cov)
W=rchisq(Sim_Number,df.opt)
R.sim = matrix(fit$center,Sim_Number,ncol(R), byrow=T) + diag( sqrt(df.opt/W) ) %*% Z # simulate
d t dist
# simulated scatterplot
pairs(R.sim,pch=16,cex=.5)
```



- `matrix(fit$center, Sim_Number, ncol(R), byrow = TRUE)` : This creates a matrix with `Sim_Number` (e.g. 5000) rows, each being a copy of the mean vector `fit$center` .

```
print(dim(matrix(fit$center,Sim_Number,ncol(R), byrow=T)))
```

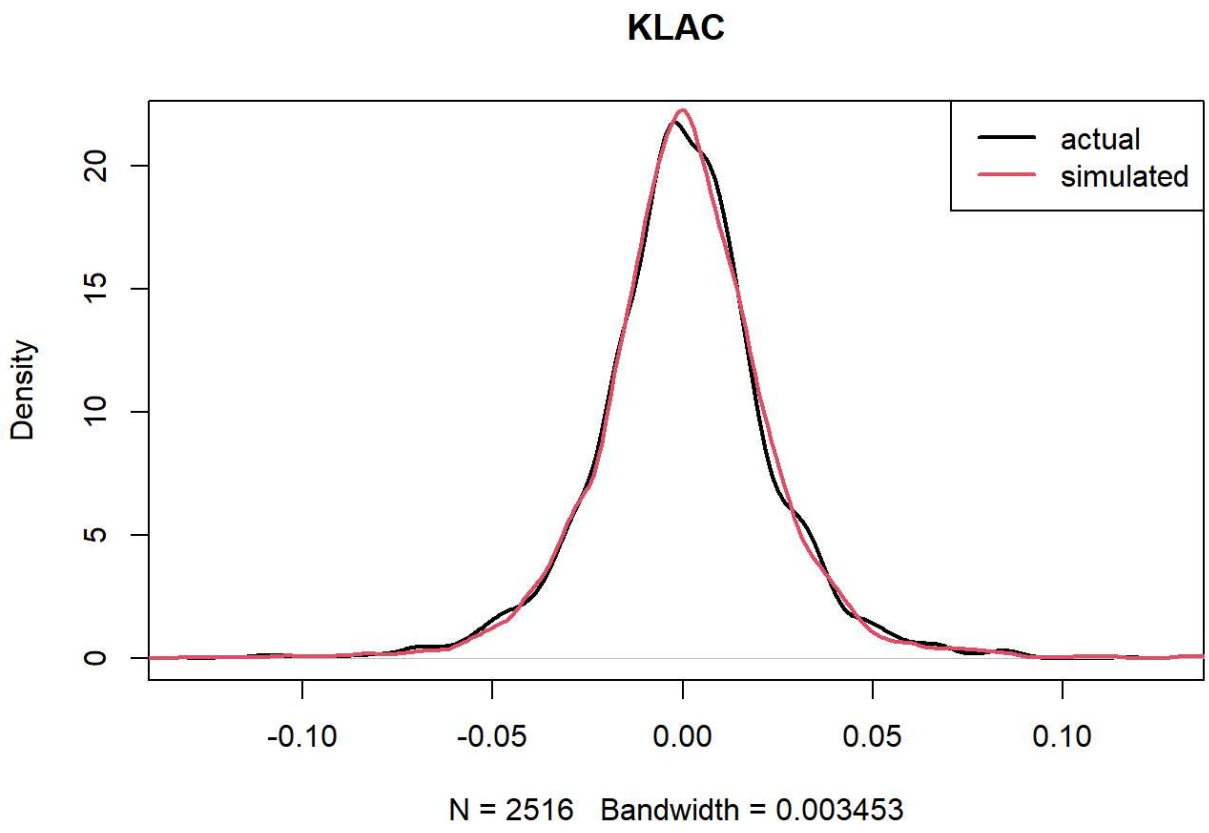
```
## [1] 5000    5
```

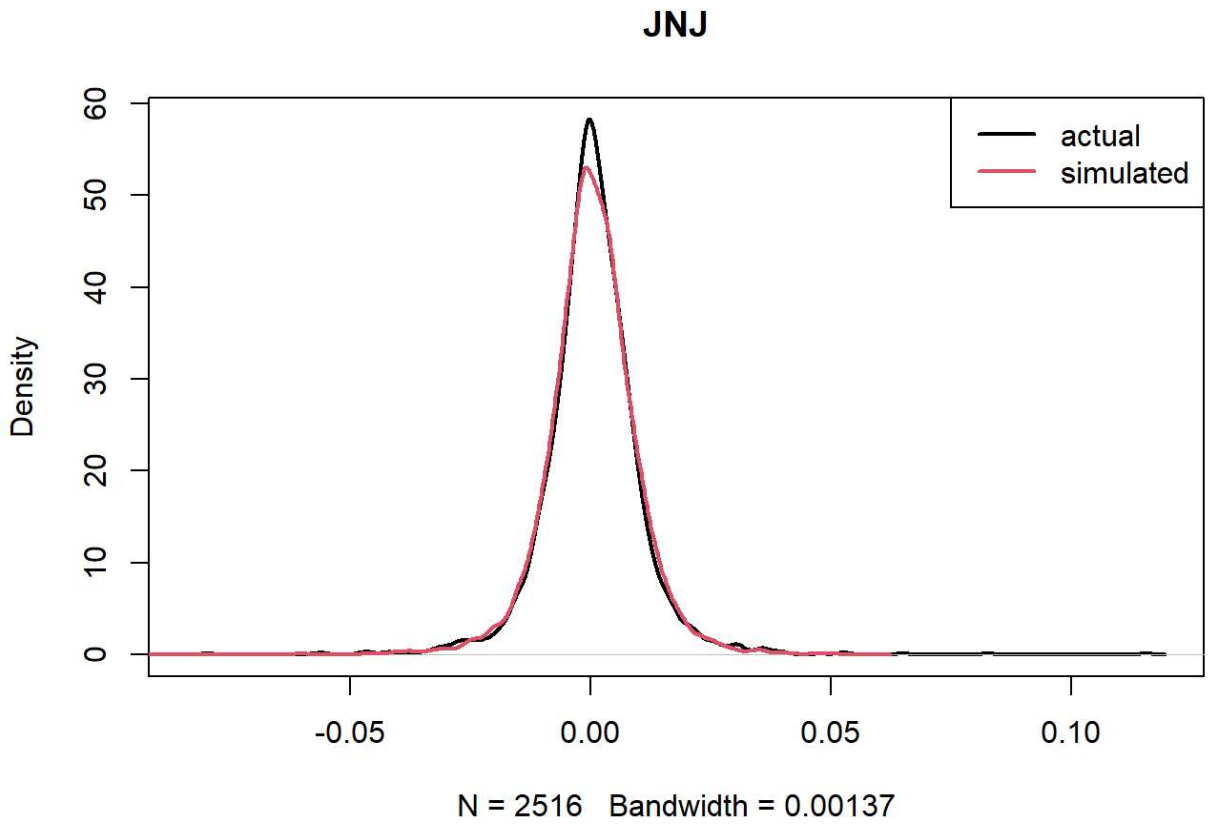
```
print(head(matrix(fit$center,Sim_Number,ncol(R), byrow=T),3))
```

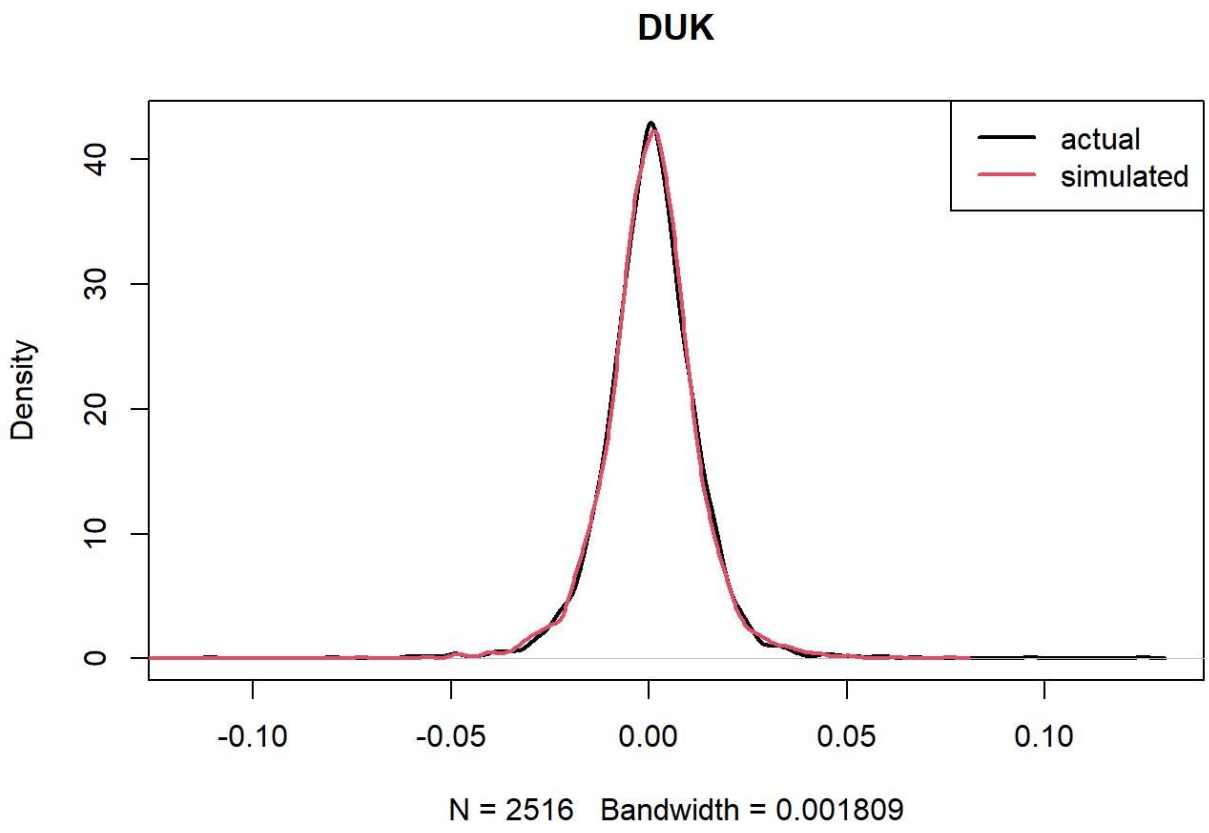
```
##              [,1]         [,2]         [,3]         [,4]       [,5]
## [1,] 0.0004105088 0.0003583425 0.0006413362 0.0003154233 0.00172763
## [2,] 0.0004105088 0.0003583425 0.0006413362 0.0003154233 0.00172763
## [3,] 0.0004105088 0.0003583425 0.0006413362 0.0003154233 0.00172763
```

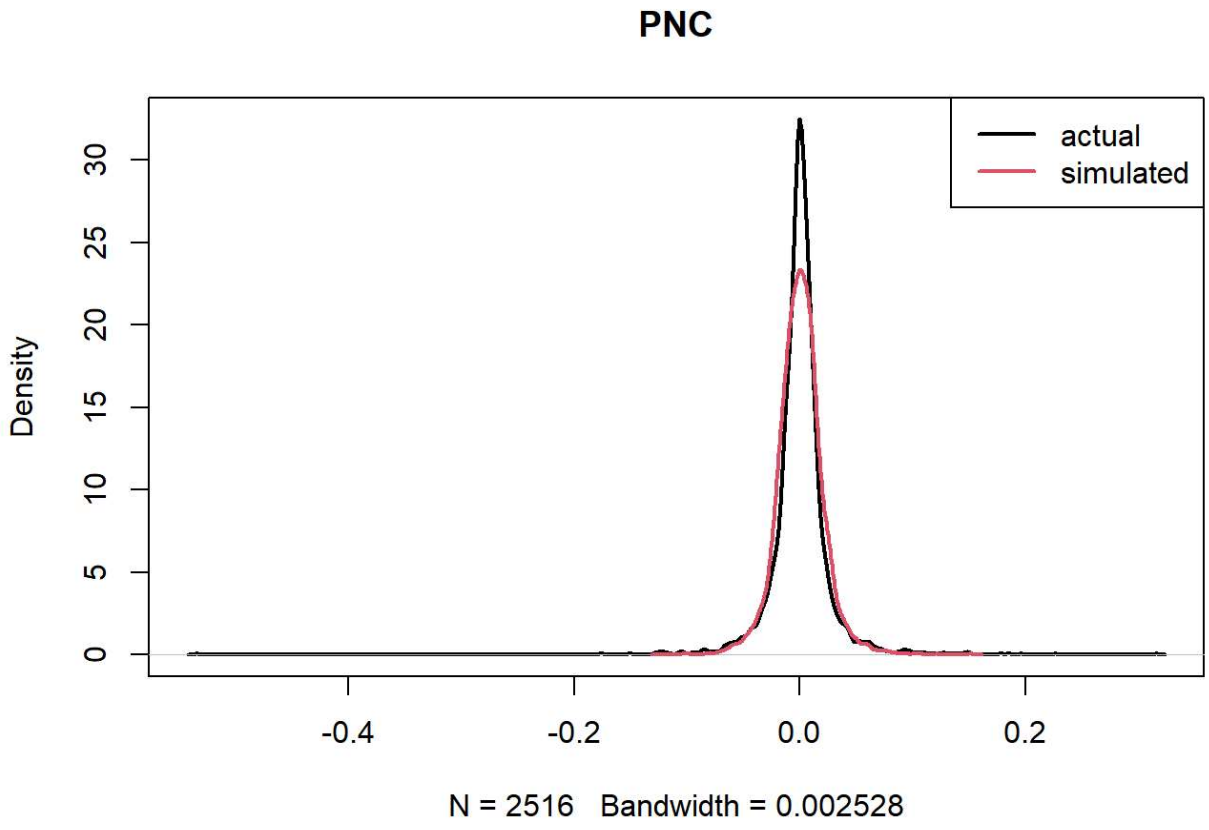
1.4.3.Compare QQ-plots of real & simulated returns

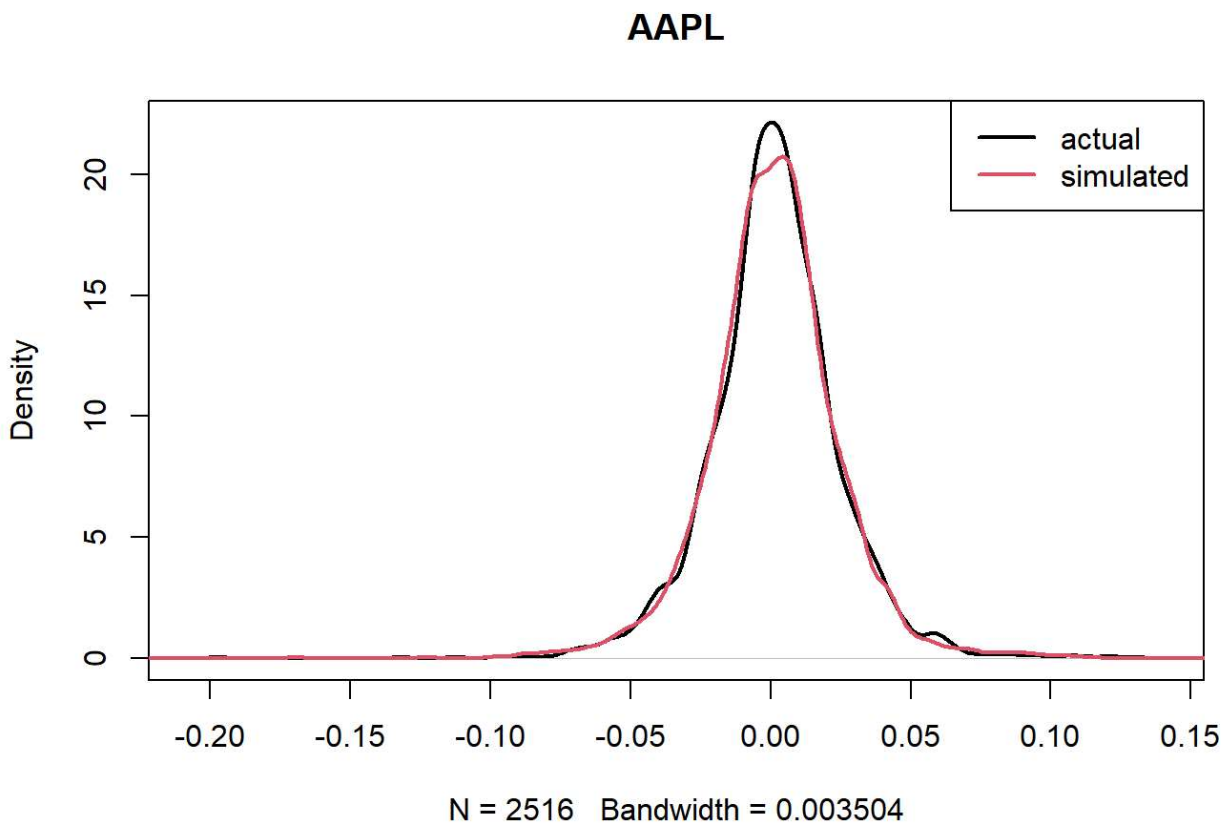
```
for(i in 1:5){
  plot(density(R[,i]), lwd=2, main=ticker.symbols[i])
  lines(density(R.sim[,i]), lwd=2, col=2)
  legend("topright", lwd=2 , col=1:2, c("actual","simulated") )
}
```


KLAC
actual
simulated
Density
0
5
10
15
20
-0.10
-0.05
0.00
0.05
0.10
N = 2516 Bandwidth = 0.003453


JNJ
actual
simulated
Density
0
10
20
30
40
50
60
-0.05
0.00
0.05
0.10
N = 2516 Bandwidth = 0.00137


DUK
actual
simulated
Density
0
10
20
30
40
-0.10
-0.05
0.00
0.05
0.10
N = 2516 Bandwidth = 0.001809


PNC
actual
simulated
Density
0
5
10
15
20
25
30
-0.4
-0.2
0.0
0.2
N = 2516 Bandwidth = 0.002528



2.Copulas

Note: in this chichi, Φ, $\Phi_{\mu,\Sigma}$ is cdf of $N(0,1)$, $N(\mu,\Sigma)$ respectively. C_ρ is guassian copula of $N(0,\rho)$.

2.1.Gaussian copula

2.1.1.Define a Gaussian copula by normal cdf, then plot contour and 3D Perspective Plot.

```
# library
library(mvtnorm) # multivariate Normal utilities
# Input: correlation matrix
rho=matrix(data=0.75,nrow=2,ncol=2); diag(rho)=1
## equivalently to:
# rho = matrix(c(1,0.75,0.75,1),nrow=2)
```

$$C_\rho(u_1,u_2)=\Phi_{0,\rho}(\Phi^{-1}(u_1),\Phi^{-1}(u_2))=pmvnorm_\rho(qnorm(u_1),qnorm(u_2))$$

- `qnorm(u_i)` : This is the inverse of the CDF of the standard normal. Transforms u from uniform(0,1) space to standard normal quantiles.

$$\Phi^{-1}(u_i)$$

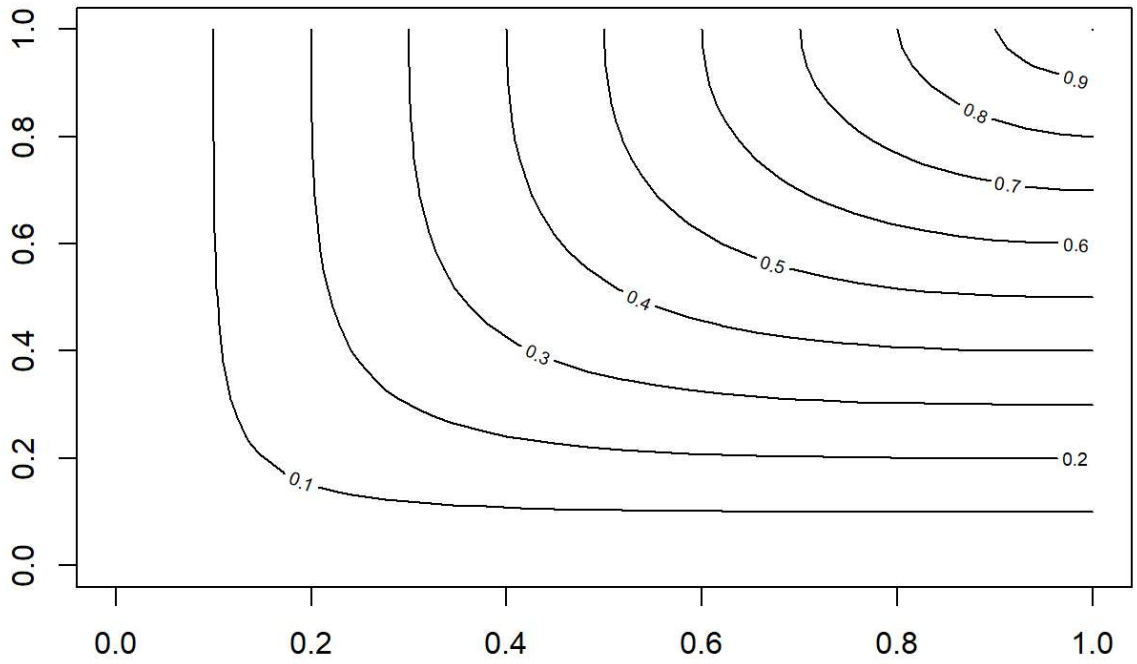
```
## define the gaussian copula = cdf of multivariate normal
GaussCopula=function(u,rho){return( pmvnorm(upper=qnorm(u),sigma=rho) )}
# pmvnorm(CDF), qnorm(inverse cdf)
```

calculate the value of copula at each point u in [0,1]*[0,1]

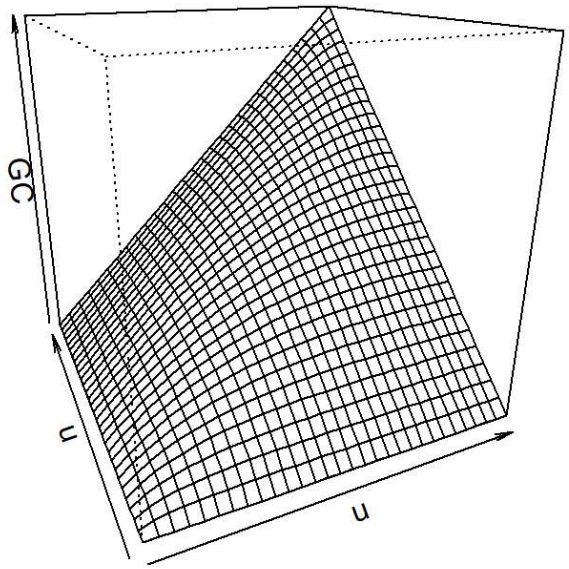
```
n=30
u=seq(0,1,length=n)
GC=matrix(0,n,n)
for(i in 1:n){
  for(j in 1:n){
    GC[i,j]=GaussCopula(c(u[i],u[j]),rho)
  }
}
```

plot contour and 3D perspective plot

```
contour(u,u,GC) # contour
```




```
persp(u,u,GC, theta=-25, phi=20) # 3D perspective plot
```



2.1.2.Generating Variates from Gaussian copula

In this part:

Step 1. we first generate Normal data $Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \sim N(0, \rho)$. Then $(U_1, U_2) = (\Phi(Z_1), \Phi(Z_2)) \sim C_\rho$.

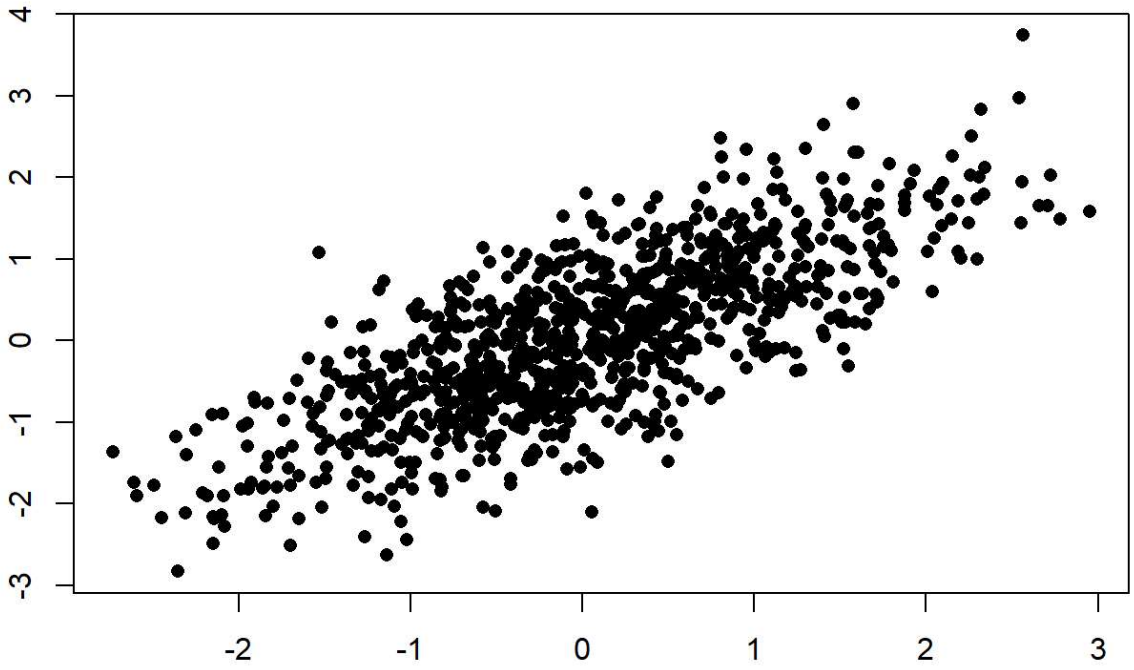
Step 2. Let F_1, F_2 be two (marginal) cdf, then $(X_1, X_2) = (F_1^{-1}(U_1), F_2^{-1}(U_2))$ has meta-Gaussian distribution with marginal F_1, F_2 and copula C_ρ.

```
#### Input: correlation, sample size
rho=matrix(data=.75,nrow=2,ncol=2); diag(rho)=1
# rho = matrix(c(1,0.75,0.75,1),nrow=2) ## equivalently
n=1000 # simulate size
```

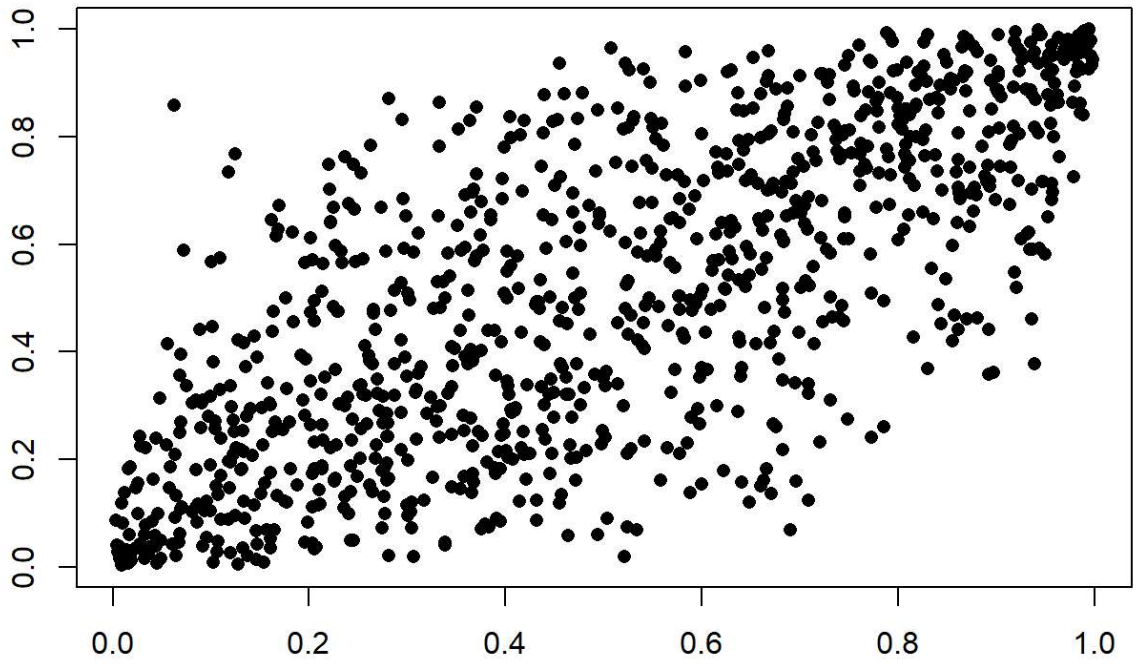
Step1.Gaussian copula variates

We first generate Normal data $Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \sim N(0, \rho)$. Then $(U_1, U_2) = (\Phi(Z_1), \Phi(Z_2)) \sim C_\rho$.

```
Z=rmvnorm(n,sigma=rho) # 2D Normal random variates
plot(Z, pch=16, xlab="",ylab="")
```




```
U=pnorm(Z) # Corresponding Gaussian copula variates
plot(U, pch=16, xlab="",ylab="")
```



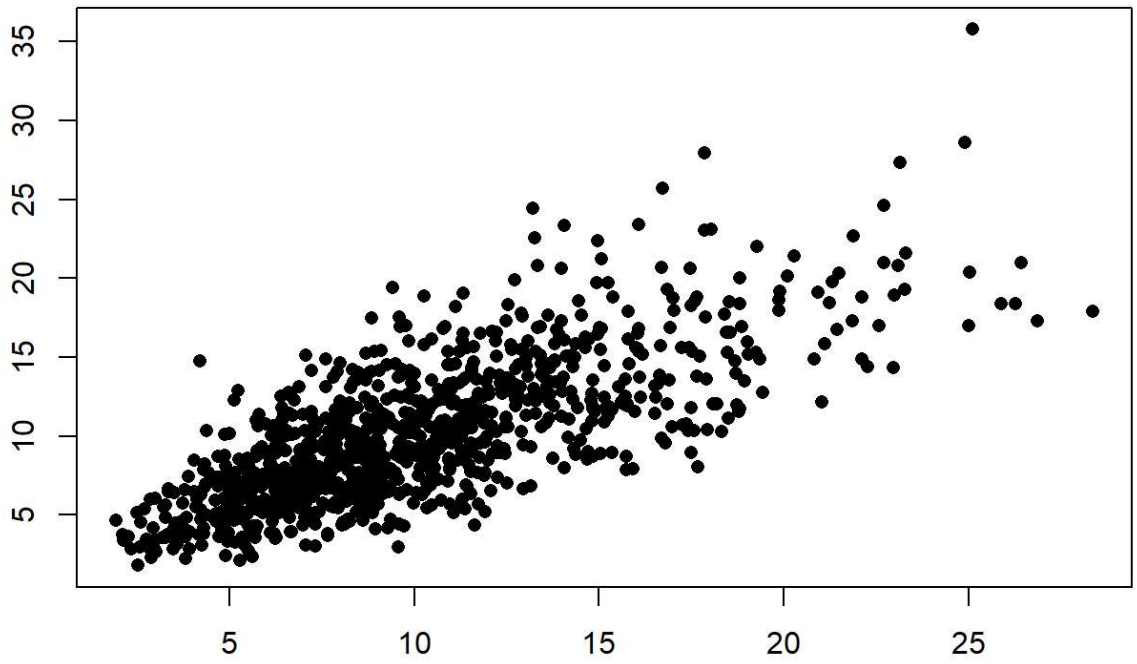
- `pnorm(Z)` → CDF (area under curve up to Z)
- `dnorm(Z)` → PDF (density at Z, i.e., height of the bell curve)

Step2.Pick Marginals get Meta-Gaussian

Let F_1, F_2 be two (marginal) cdf, then $(X_1, X_2) = (F_1^{-1}(U_1), F_2^{-1}(U_2))$ has meta-Gaussian distribution with marginal F_1, F_2 and copula C_ρ.

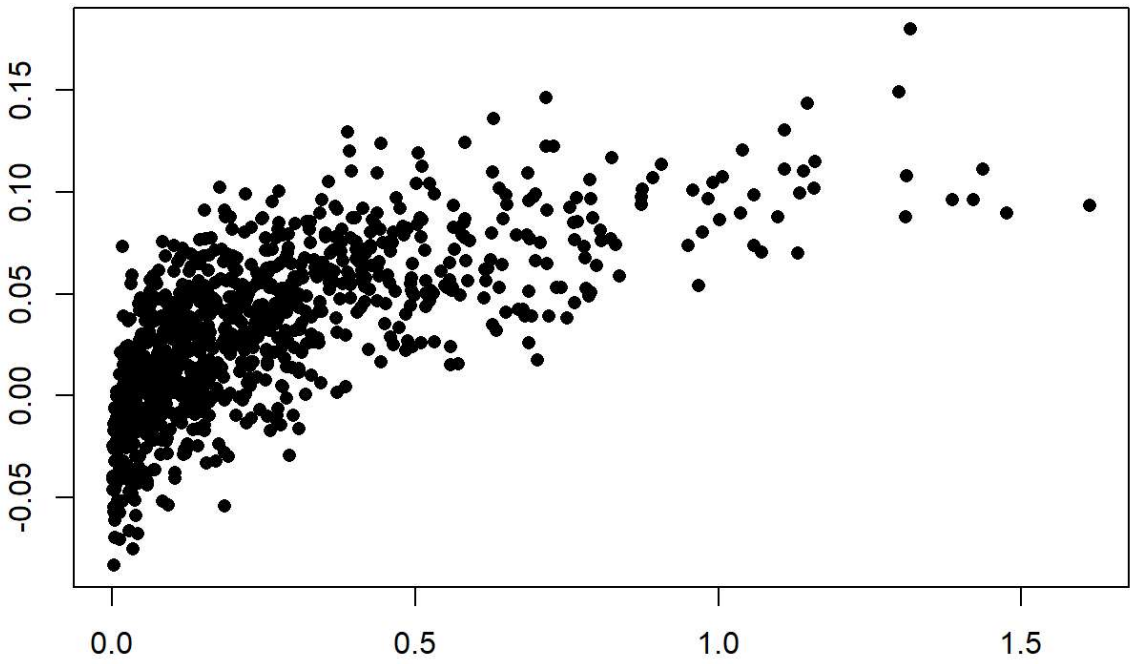
step2.eg1.Meta-Gaussian with both marginal $\chi^2(df = 10)$

```
X=qchisq(U,10) # Generating (meta-Gaussian) Chi-square variates
plot(X, pch=16, xlab="",ylab="")
```



step2.eg2.Meta-Gaussian with marginal exponential & normal

```
X1=qexp(U[,1],4) # Generating (meta-Gaussian) Exponential & Normal variates
X2=qnorm(U[,2],.03,.04)
plot(X1,X2, pch=16, xlab="",ylab="")
```



2.2.Elliptical copulas (Gaussian copula and t copula)

Gaussian copula only depends on ρ.

t copula depends on ρ and df.

```
library(copula)
```

2.2.1 p2P and P2p (Correlation vector)

In `normalCopula()` and `tCopula()`, the input correlation should be a **vector** instead of correlation matrix.

Vector of correlations => correlation matrix

```
rho_vector = c(.1,.2,.3)
rho_mat=p2P(rho_vector)
rho_mat
```

```
##      [,1] [,2] [,3]
## [1,]  1.0  0.1  0.2
## [2,]  0.1  1.0  0.3
## [3,]  0.2  0.3  1.0
```

```
p2P(0.6)
```

```
##      [,1] [,2]
## [1,]  1.0  0.6
## [2,]  0.6  1.0
```

Correlation matrix => Vector

```
rho_mat = matrix(c(1.0, 0.1, 0.2, 0.1,1.0,0.3,0.2,0.3,1.0), 3, 3)
P2p(rho_mat)
```

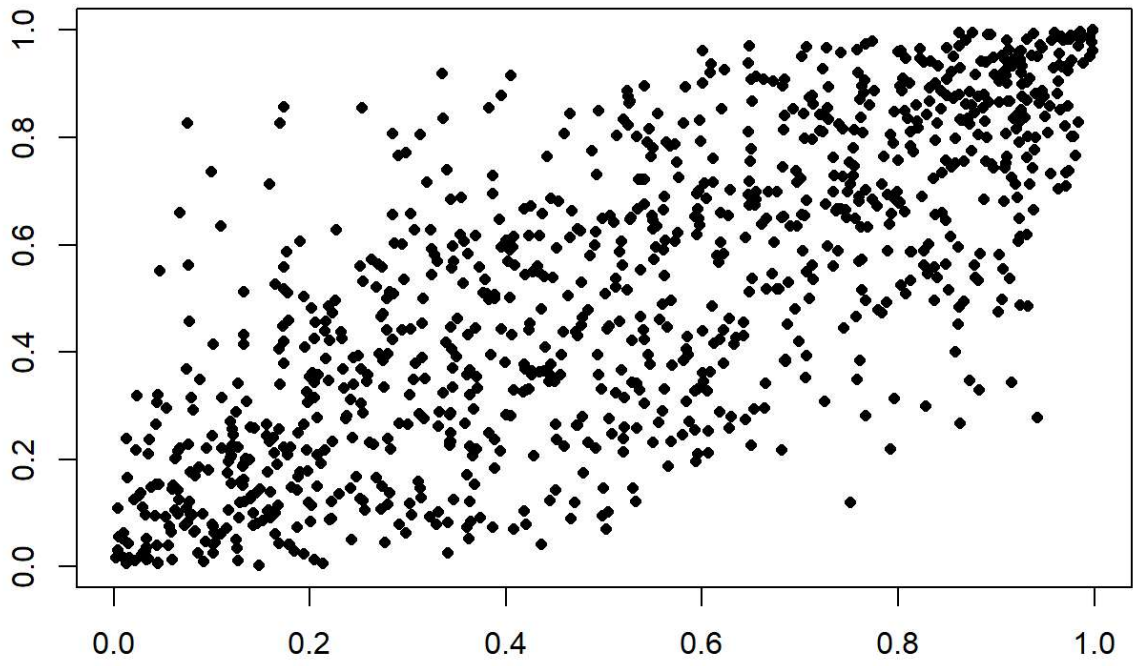
```
## [1] 0.1 0.2 0.3
```

2.2.2.Normal copula (Gaussian copula use library)

```
## Input: correlation vector (library need),sample size
rho_vector = 0.75
n=1000
```

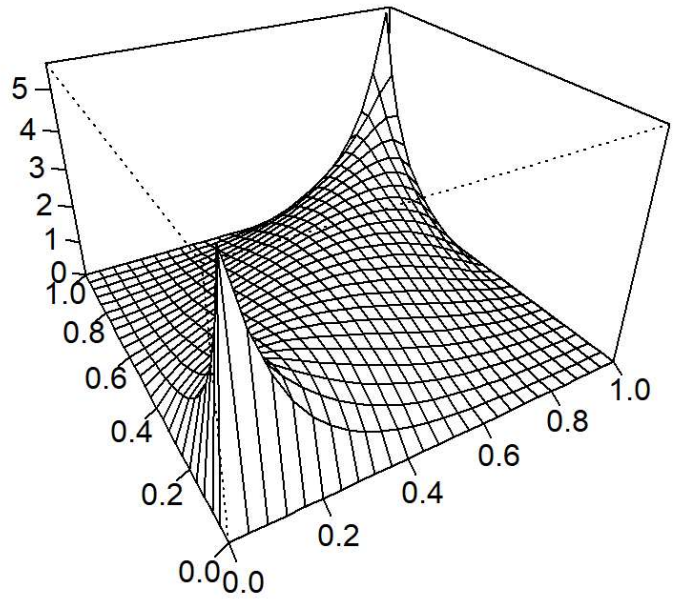
(a) Construct normal copula

```
Normal.cop = normalCopula( rho_vector, dim=2, dispstr="un")
u = rCopula(n, Normal.cop)
plot(u, pch=16, cex=.8, xlab="", ylab="")
```

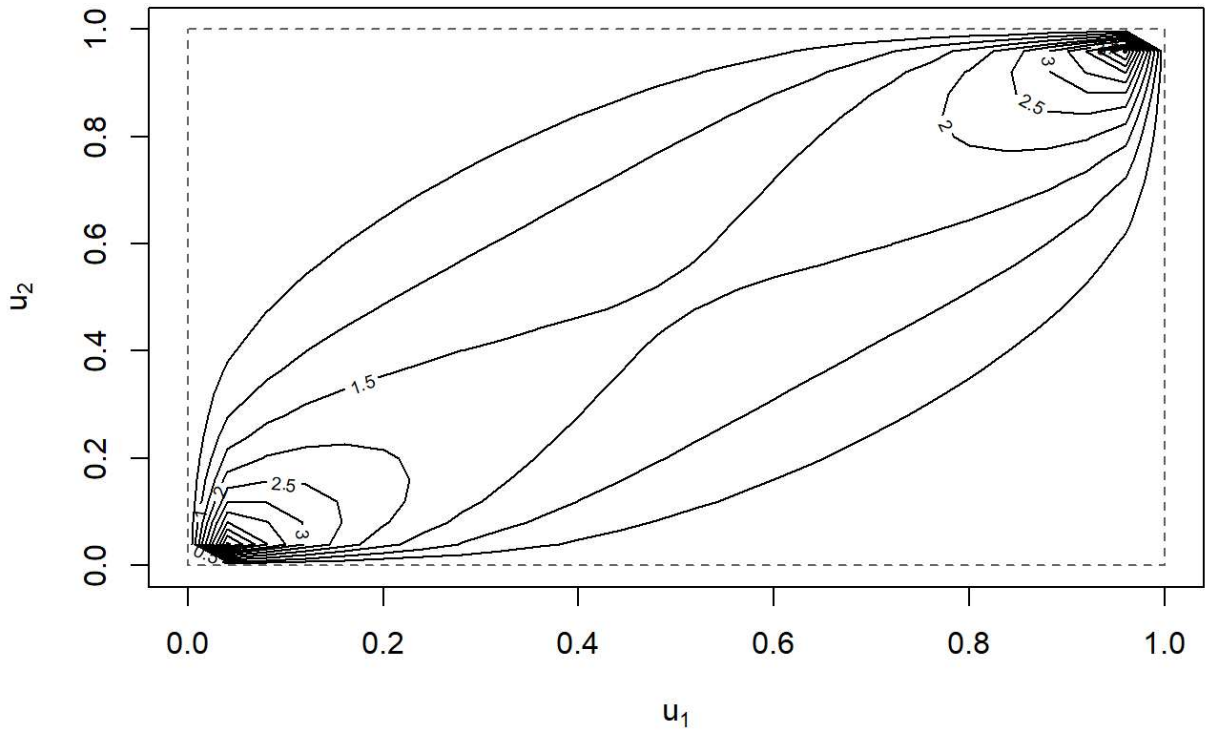


3D plot (pdf), pdf and cdf contour plots

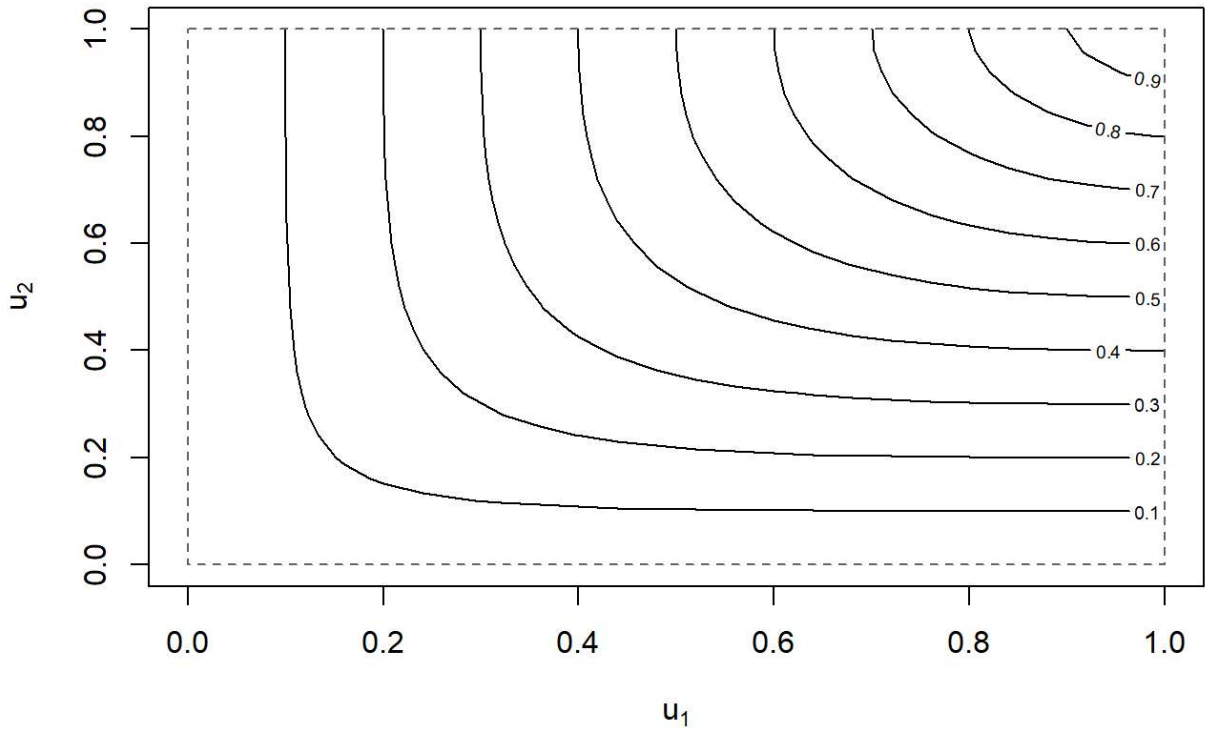
```
persp(Normal.cop,dCopula, zlab="", xlab="", ylab="") # plot of copula pdf
```




```
contour(Normal.cop,dCopula) # contour of copula pdf
```




```
contour(Normal.cop,pCopula) # contour of copula cdf
```



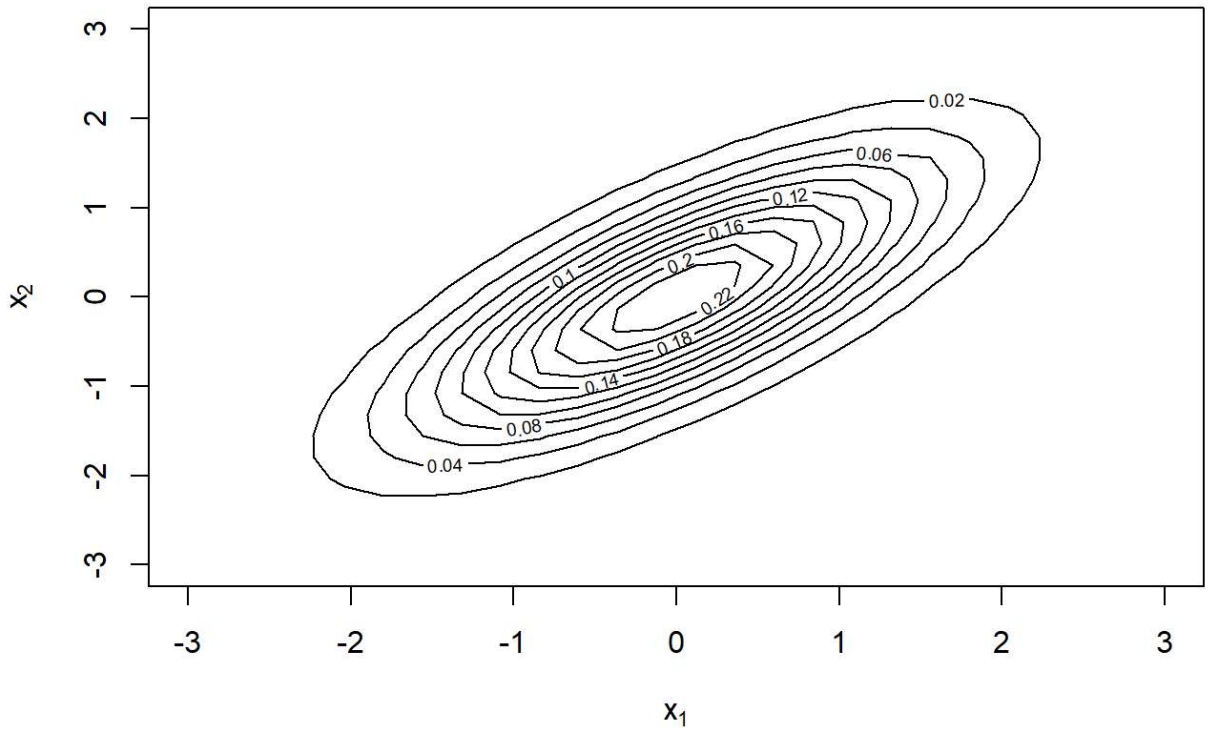
(b) Generate meta gaussian

e.g.1.Bivariate normal: normal copula and marginal N(0,1)

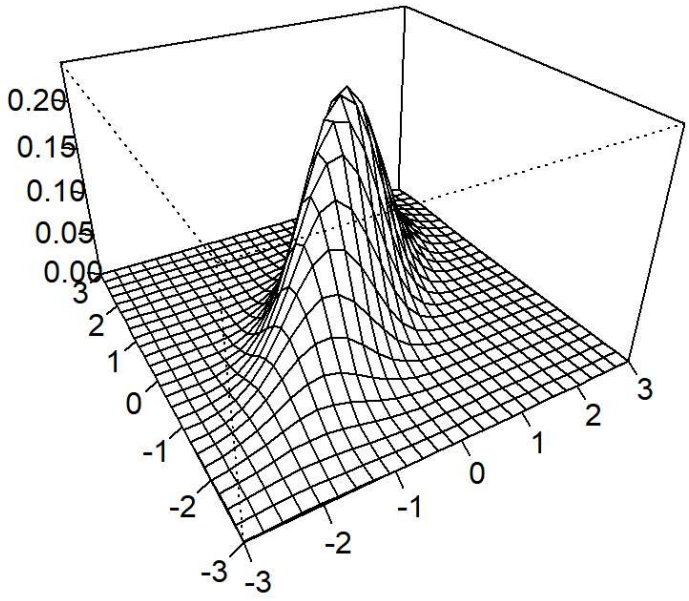
```
Normal.Mvd = mvdc(Normal.cop, margins = c("norm", "norm"),
              paramMargins = list(list(mean = 0, sd = 1), list(mean = 0, sd = 1)))
```

bivariate normal pdf 3D plot and contour plot

```
contour(Normal.Mvd,dMvdc, xlim = c(-3, 3), ylim = c(-3, 3)) # contours of bivariate density
```




```
persp(Normal.Mvd,dMvdc, xlim = c(-3, 3), ylim = c(-3, 3), zlab="", xlab="", ylab="") # 3D plot o
f bivariate density
```



e.g.2.Exponential marginal meta-Gaussian data

Gives two exponential variables, each with their own rate, but introduce dependency between them via a Gaussian copula. Copulas let you separate marginals from dependency structure.

- `Normal.cop` : The copula defines the **dependence structure** (correlation between variables).
- `margins = rep("exp", 2)` : You want two **Exponential marginals**.
- `paramMargins` : Each marginal has its own rate:
  - First variable: Exponential with rate 2
  - Second variable: Exponential with rate 3

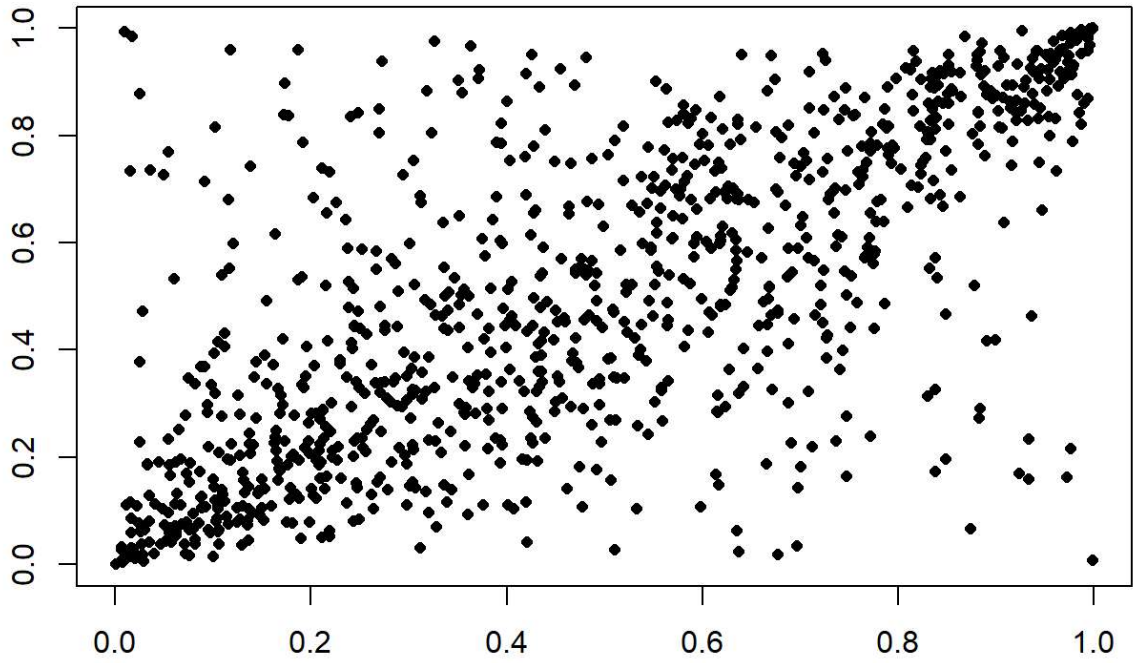
```
Exp.Mvd = mvdc(Normal.cop, margins =  rep("exp",2),
                   paramMargins = list(list(rate=2), list(rate=3)))
Exp.sample=rMvdc(n = 1000, mvdc = Exp.Mvd)
```

2.2.3.t copula

```
## Input: correlation vector,sample size,
rho_vector = 0.75
n=1000
df=2
```

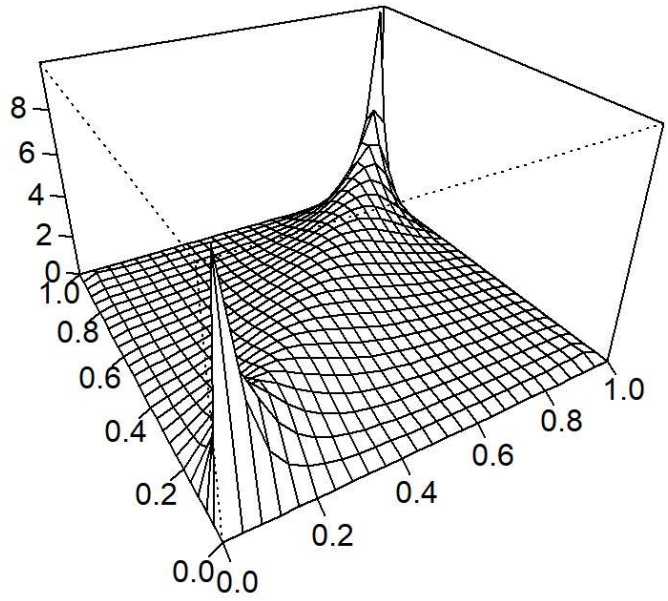
(a) construct t copula, generating and plot sample data from copula

```
t.cop = tCopula( rho_vector, df=df, dim = 2, dispstr="un") # 2D Student-t copula with df=2 & 0.7
5 correlation
u = rCopula(n, t.cop)
plot(u, pch=16, cex=.8, xlab="", ylab="")
```

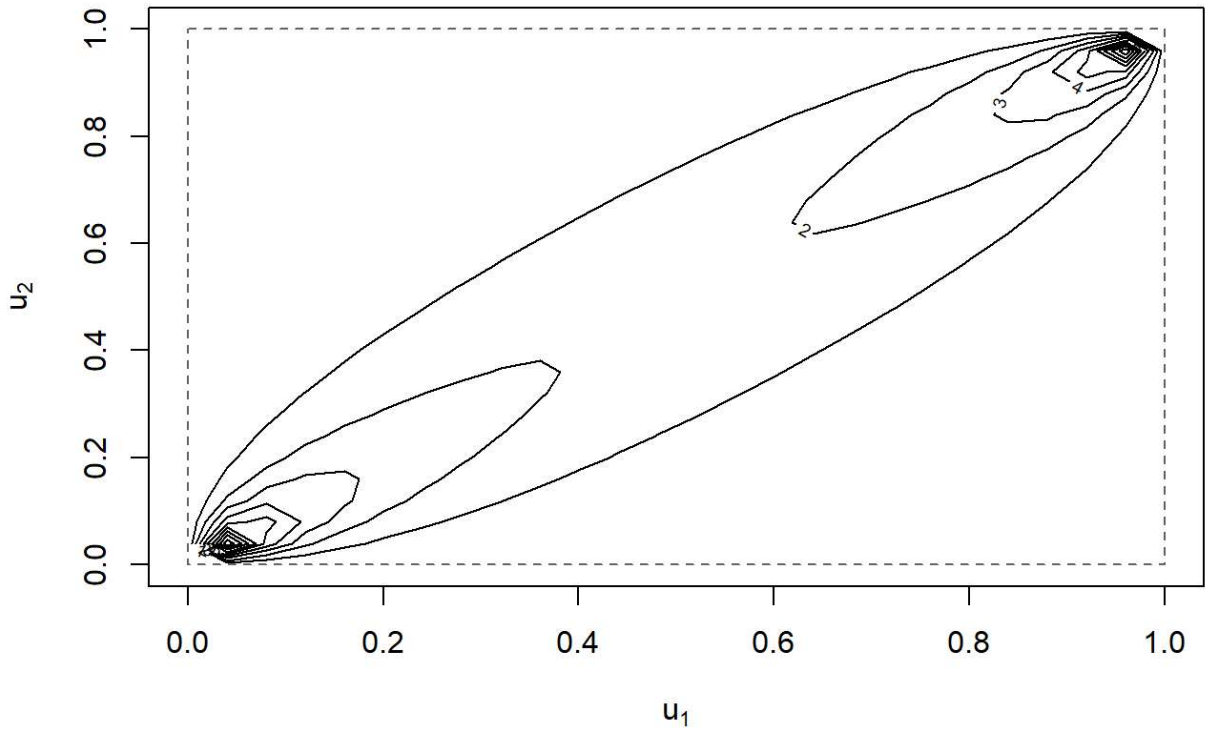


copula pdf 3D plot, pdf contour plot

```
persp(t.cop,dCopula, zlab="", xlab="", ylab="")
```



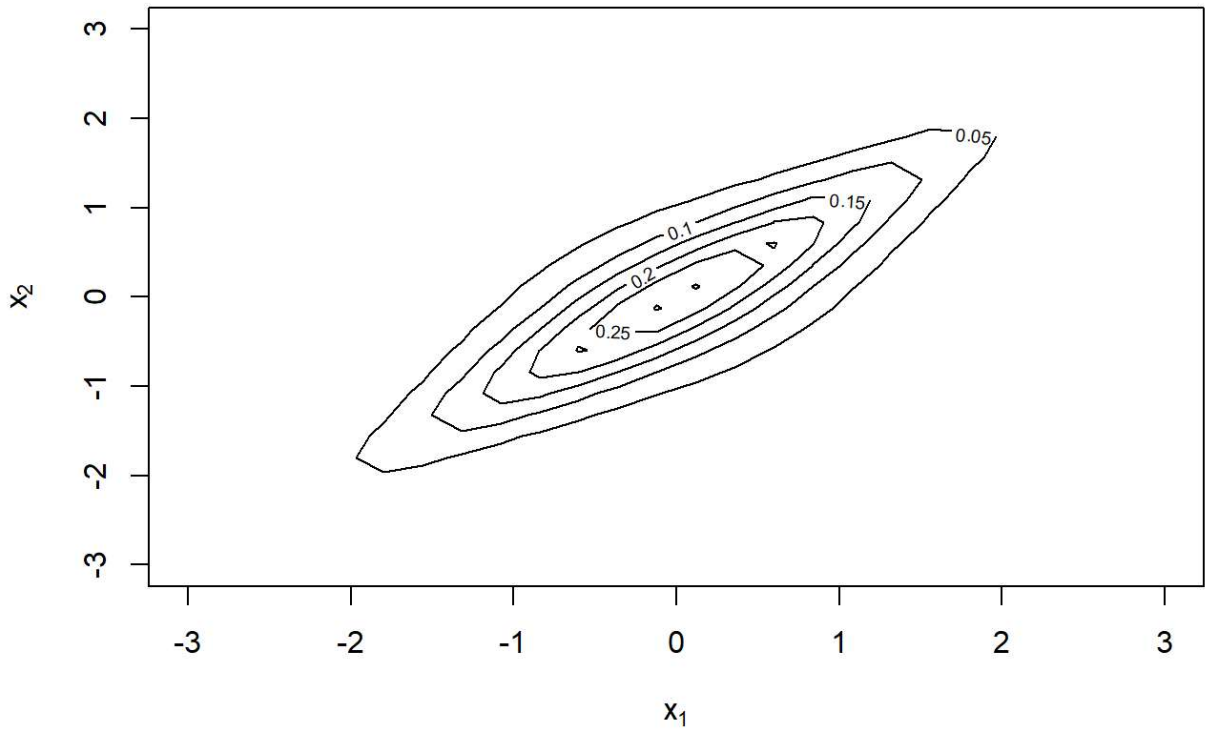

```
contour(t.cop,dCopula)
```



(b) Generate multivariate distribution with t copula

e.g.1.Multivariate distribution with t copula & N(0,1) marginals, plot contour of pdf

```
t.Mvd = mvdc(t.cop,margins = c("norm", "norm"),
             paramMargins = list(list(mean = 0, sd = 1), list(mean = 0, sd = 1)))
contour(t.Mvd,dMvdc, xlim = c(-3, 3), ylim = c(-3, 3))
```



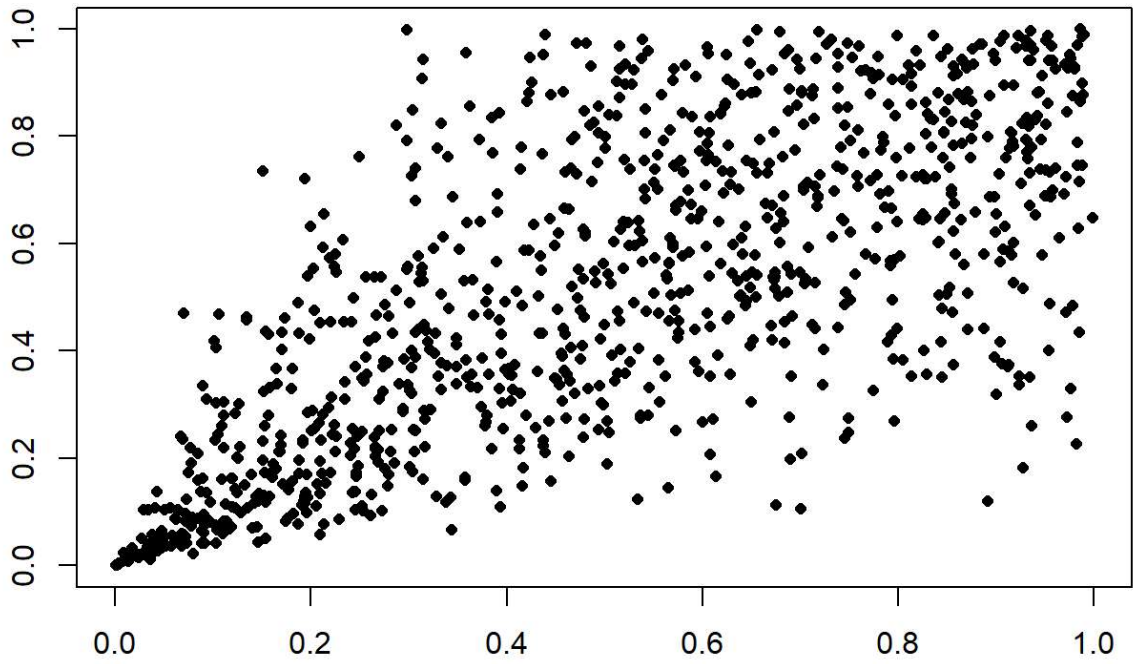
2.3.Archimedean copulas

2.3.1.Clayton/Frank/Gumbel copula

```
## Input: copula,sample size
n=1000

## Clayton
cop = claytonCopula(param=2.5, dim=2)
## or Frank
# cop = frankCopula(param=9, dim=2)
## or Gumbel
# cop = gumbelCopula(param=2.5, dim=2)
```

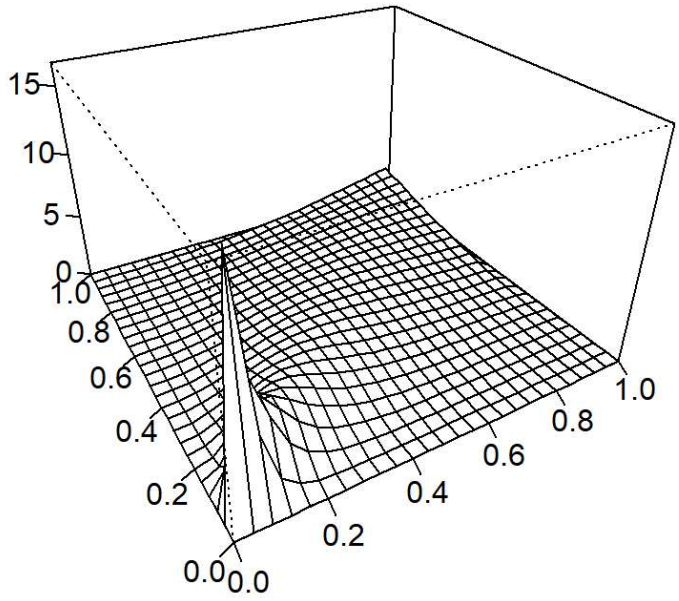
generating data from copula and plot

```
u = rCopula(n, cop)
plot(u, pch=16, cex=.8, xlab="", ylab="")
```

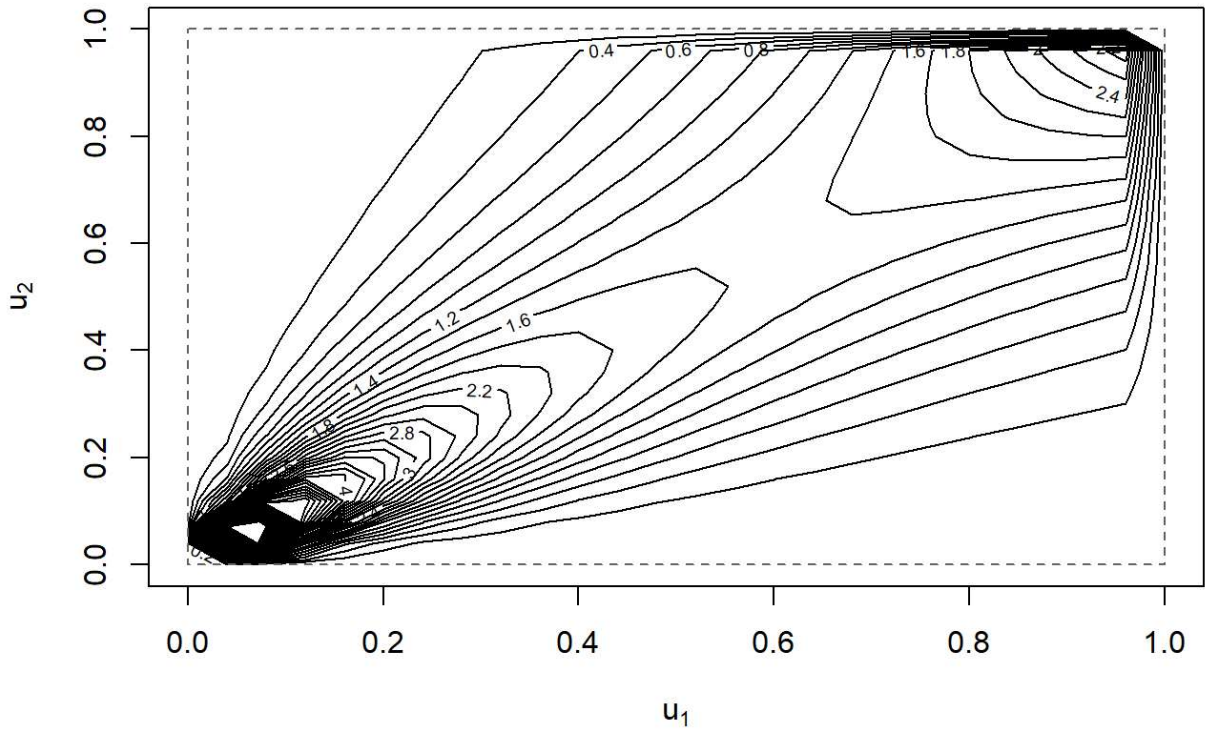


pdf 3D perspective plot, pdf and cdf contour plots

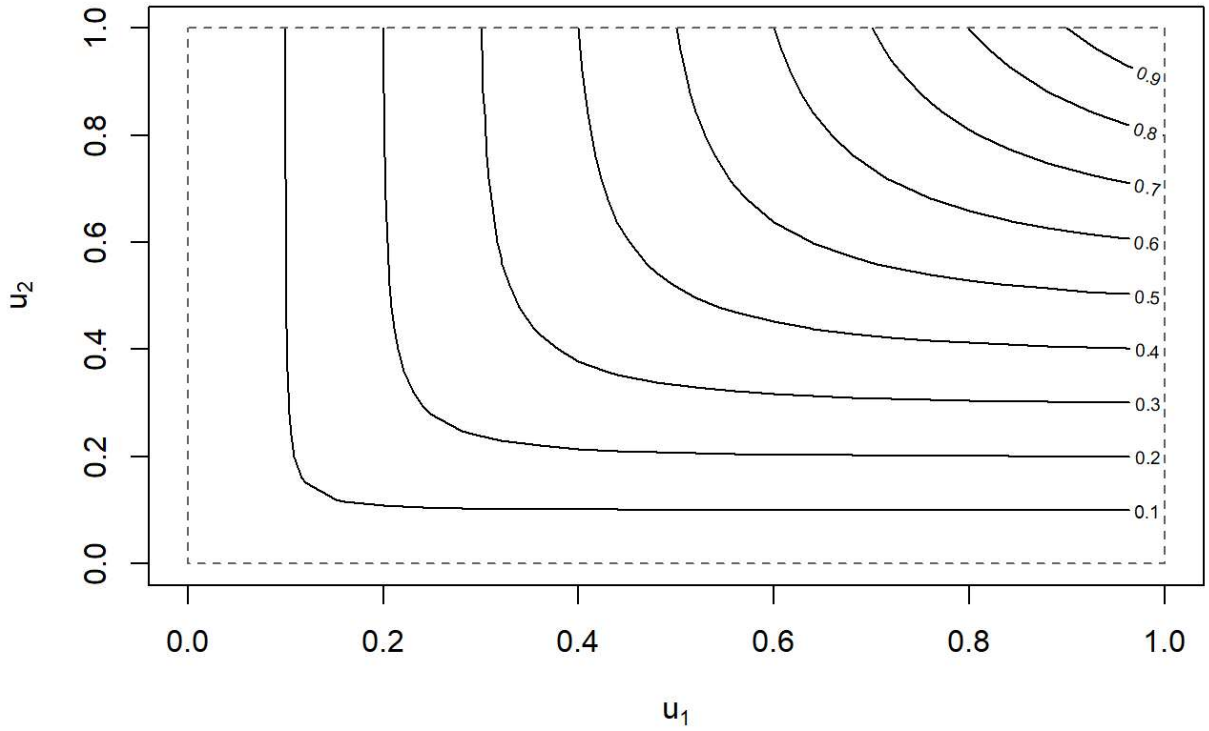
```
persp(cop,dCopula, zlab="", xlab="", ylab="")
```




```
contour(cop,dCopula, nlevels=100)
```

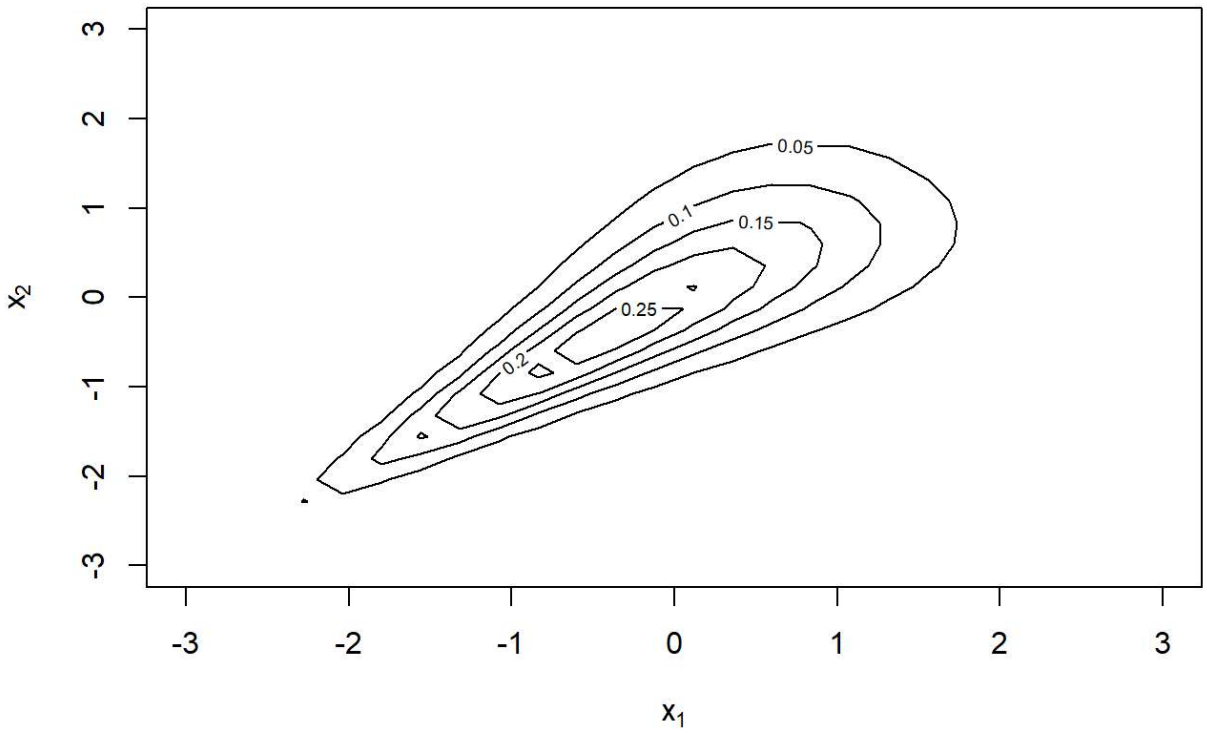


```
contour(cop,pCopula)
```



2.3.2.Generate multivariate distribution with chosen (Clayton/Frank/Gumbel) copula & N(0,1) marginals, plot contour of pdf

```
Mvd = mvdc(cop, margins = c("norm", "norm"),
           paramMargins = list(list(mean = 0, sd = 1), list(mean = 0, sd = 1)))
contour(Mvd, dMvdc, xlim = c(-3, 3), ylim = c(-3, 3))
```


0.05
0.1
0.15
0.2
0.25
x2
x1
-3
-2
-1
0
1
2
3