```
#importation library
library(nortest)
library(Dowd)
library(corrplot)
library(PerformanceAnalytics)
library(timeSeries)
library(ggplot2)
library(MASS)
library('timeDate')
library(LambertW)
library("QRM")
library(copula)
library(fCopulae)
library (LambertW)
library("limma")
library('asbio')
library(VineCopula)
library(lcopula)
##fonction de calcul des log-rendements
logrendement=function(v){
  n= length(v)
  u=vector ()
  for (i in 1: n)
  {if (i==1)
    u[i]=0
  else
    u[i]=log(v[i]/v[i-1])}
  return(u)}
# Fonction signe :
signe=function(x){if(x>=0) {k=1}else k=-1}
# fonction de comptage sur une condition donnée
rang=function(x,y) {
  rank=NULL
  for (i in 1:length(x)) {
    c=0
    for (j in 1:length(x)) {
      if (isTRUE((j!=i)) & isTRUE((y[j]<=y[i])) & isTRUE(x[j]<=x[i]))
{c=c+1}
    }
    rank[i]=c
  }
  return(rank)
}
#importation de la base de donnée
BCP=read.csv('BCP.csv',header=TRUE,sep=',')
colnames(BCP)=c('date','price_BCP','A','B','C','D','E')
BCP=subset(BCP, select=-c(A:E))

BOA=read.csv('BOA.csv',header=TRUE,sep=',')
colnames(BOA)=c('date','price_BOA','A','B','C','D','E')
BOA=subset(BOA, select=-c(A:E))
#fusionner les deux bases by id=date
BASE=merge(BCP,BOA,by='date')
#CALCUL LOG RENDEMENT
BOA_R=logrendement(BASE$price_BOA)
BCP_R=logrendement(BASE$price_BCP)

#Visulaiser les deux series
```

```
par(mfrow=c(2,1))
plot(BASE$price_BCP,main='BCP',type='l',xlab='date',ylab='Prix de
fermeture',col='red')
plot(BASE$price_BOA,main='BOA',type='l',xlab='date',ylab='Prix de
fermeture',col='blue')

## tracé du graphe des log-rendements
u=BCP_R
v=BOA_R
u.ts=ts(u, start = c (2005,11,22), end = c (2021,12,21), frequency = 365)
plot.ts(main="les log-rendements du BCP", u.ts, xlab="temps", ylab="log-
rendement",col='red')
v.ts=ts(u, start = c (2005,11,22), end = c (2021,12,21), frequency = 365)
plot.ts(main="les log-rendement BOA", v.ts,xlab="temps", ylab="log-
rendement",col="blue")
## statistiques descriptive #
summary(BASE$price_BCP)
summary(BASE$price_BOA)
#Ecart type
sd(BASE$price_BCP)
sd(BASE$price_BOA)
#calibrage_des fnct de repartition
x= BOA_R
a=fitdistr(x,'normal')$estimate
hist(x, freq = FALSE,main='Histogramme et densite pour  les rendements
BOA')
curve(dnorm(x, a[1], a[2]), add = TRUE , col='red')

y= BCP_R
b=fitdistr(y,'normal')$estimate
hist(y, freq = FALSE,main='Histogramme et densite pour  les rendements
BCP')
curve(dnorm(x, b[1], b[2]), add = TRUE,col='red')
#testes
#1_NORMALITE
#kolmogrov test
ks.test(BCP_R,"pnorm",mean=mean(BCP_R), sd=sd(BCP_R))
ks.test(BOA_R,"pnorm",mean=mean(BOA_R), sd=sd(BOA_R))
ks.test(BCP_R,"pt",df=n-1)
ks.test(BOA_R,"pt",df=n-1)

#shapiro test
shapiro.test(BCP_R)
shapiro.test(BOA_R)
#lillie test
lillie.test(BCP_R)
lillie.test(BOA_R)
#qqplot
qqnorm(BCP_R)
qqline(BCP_R, distribution = qnorm,col='red')
qqnorm(BOA_R)
qqline(BOA_R, distribution = qnorm,col='blue')
#tqqplot
par(mfrow=c(1,2))
n=length(BCP_R)
TQQPlot(BCP_R,df=4)
TQQPlot(BOA_R,df=4)
#detection dependance
```

```
#1_avec corr plot
D=cbind(BCP_R,BOA_R)
chart.Correlation(D, histogram=TRUE)
# Diagramme de dispersion
plot(x = BCP_R,y = BOA_R,main="Diagramme de dispersion des rendements
logarithmiques")
# le  RANK RANK plot
u=rang(x = BCP_R,y = BCP_R)
v=rang(x = BOA_R,y =BOA_R)
plot(x = u,y = v,main="Diagramme Rank-Rank")
#visulaiser la correlation
ggplot(BASE,aes(x=BCP_R,BOA_R))+geom_point()
#khi plot
chi.plot(BCP_R,BOA_R, main='KHI PLOT')
#k plot
BiCopKPlot(pobs(BCP_R),pobs(BOA_R),main='K plot du BCP et BOA')
# Estimation de tau de kendall
estim_tau_kend=function(x,y){
  k=0
  for (j in 2:n){
    c=0
    for (i in 1:j-1) {
      if(isTRUE((x[j]-x[i])*(y[j]-y[i])>=0)) {s=1}else s=-1
      c=c+s
    }
    k=k+c
  }
  return((2/(n*(n-1)))*k)}
##########################
#estimation parametre avec kandall inverse
estim_para=function(char,x,y) {

  tau_kendell=estim_tau_kend(x,y)
  rho=cor(x,y)
  k=1
  if (char=='Gumbel') {k=1/(1-tau_kendell)}
  if (char=='Clayton') {k=2*tau_kendell/(1-tau_kendell)}
  if (char=='FGM') {k=9*tau_kendell/2}
  if (char=='Gaussienne'  & rho==1) {k=(2/pi)*asin(rho)}
  if (char=='Gaussienne'  & rho<0) {k=(2/pi)*asin(rho)}
  if (char=='Student') {k=sin(tau_kendell*pi/2)}

  return(k)
}
D=data.frame(Copule=0:0,Gumbel=0:0,Clayton=0:0,FGM=0:0,Gaussienne=0:0,Student=0:0,FRANK=
D$Copule="parametre"
D$Gumbel=estim_para('Gumbel',BCP_R,BOA_R)
D$Clayton=estim_para('Clayton',BCP_R,BOA_R)
D$FGM=estim_para('FGM',BCP_R,BOA_R)
D$Gaussienne=estim_para('Gaussienne',BCP_R,BOA_R)
D$Student=estim_para('Student',BCP_R,BOA_R)
D$FRANK=3.03
# Ajustement - Copule gaussienne#
ro=cor(BCP_R,BOA_R)
B=cbind(BCP_R,BOA_R)
Udata=pobs(B)
norm.cop <- normalCopula (ro, dim = 2, dispstr = "un")
norm.cop #Informations sur l'objet
```

```
NormCopEst<-fitCopula (norm.cop, Udata, method="mpl")
NormCopEst
logLik (NormCopEst)
AIC(NormCopEst)
BIC(NormCopEst)

# Ajustement - Copule t-Student #
tCop <- tCopula (ro, dim = 2, dispstr="un", df.fixed=FALSE)
tCop
TCopEst<-fitCopula (tCop, Udata, method="mpl",estimât.variance=TRUE)
TCopEst
logLik(TCopEst)
AIC(TCopEst)
BIC(TCopEst)
##########################
# Ajustement - Copule de Gumbel #
ParGum=estim_para('Gumbel',BCP_R,BOA_R)
gumb.cop0 <- gumbelCopula (ParGum, dim =2)
gumb.cop0
gumbCopEst<-fitCopula (gumb.cop0, Udata, method="mpl")
logLik(gumbCopEst)
AIC (gumbCopEst)
BIC (gumbCopEst)
##############################
# Ajustement - Copule de Clayton #
ParClay=estim_para('Clayton',BCP_R,BOA_R)
clay.cop0<- claytonCopula (param =ParClay, dim = 2)
clay.cop0
ClayCopEst<-fitCopula (clay.cop0, Udata, method="mpl")
logLik (ClayCopEst)
AIC(ClayCopEst)
BIC(ClayCopEst)
############################

# Ajustement - Copule de Frank
frank.cop0<-frankCopula (param = NA_real_, dim = 2)
frank.cop0
FrankCopEst<-fitCopula (frank.cop0, Udata, method="mpl")
FrankCopEst
logLik(FrankCopEst)
AIC(FrankCopEst)
BIC(FrankCopEst)
# Ajustement - Copule de FGM
fgm.cop0<-fgmCopula (param = NA_real_, dim = 2)
fgm.cop0
FGMCopEst<-fitCopula (frank.cop0, Udata, method="mpl")
FGMCopEst
logLik(FGMCopEst)
AIC(FGMCopEst)
BIC(FGMCopEst)

B=data.frame(valeur=0:2,Gaussienne=0:2,Student=0:2,FGM=0:2,FRANK=0:2,Clayton=0:2,Gumbel
B$valeur=c('logLik','AIC','BIC')
B$Gaussienne=c(logLik (NormCopEst),AIC(NormCopEst),BIC(NormCopEst))
B$FGM=c(logLik(FGMCopEst),AIC(FGMCopEst),BIC(FGMCopEst))
B$FRANK=c(logLik(FrankCopEst), AIC(FrankCopEst),BIC(FrankCopEst))
B$Gumbel=c(logLik(ClayCopEst),AIC(ClayCopEst),BIC(ClayCopEst))
B$Clayton=c(logLik(gumbCopEst),AIC(gumbCopEst),BIC(gumbCopEst))
```

```
B$Student=c(logLik(TCopEst),AIC(TCopEst),BIC(TCopEst))
B$best=c('-',"Gumbel","Gumbel")

###########################
#copule archim

x=BCP_R
y=BOA_R
#clayton
clayton_cop=function(u,v){
  theta=estim_para(char = 'Clayton',x = x,y = y)
  return((u^(-theta)+v^(-theta)-1)^(-1/theta))
}
# Frank :
Frank_cop=function(u,v){
  theta=estim_para(char = "Frank",x = x,y = y)
  return(-(1/theta)*log(1+((exp(-theta*u)-1)*(exp(-theta*v)-1)/((exp(-
theta)-1)))))
}
#Gumbel
Gumbel_cop=function(u,v){
  theta=estim_para(char = 'Gumbel',x = x,y = y)
  result=exp(-((-log(u))^(theta)+(-log(v))^theta)^(1/theta))
  return(result)
}
# Copules elliptiques :
# Student :
Student_cop=function(u,v,nu){
  # nu est l'estimateur du degré de liberté
  # u et v sont uniformémenet distribués
  tau_kend=estim_tau_kend(u,v)
  k=floor(nu)
  rho=sin(tau_kend*pi/2)
  value=(k/(2*sqrt(1-rho^2)))*((gamma(k/2))^2/(gamma((k+1)/2))^2)*((1+
((u^2+v^2-2*rho*u*v)/(k*(1-rho^2))))^(-(k+2)/2))/((1+(1/k)*u^2)*(1+(1/
k)*v^2))^(-(k+2)/2)
  return(value)}
# ce sont des copules estimées pour les theta estimées

#  Maintenant implémentation de la copule empirique de Deheuvels dont on va
calculer l'écart et conclure pour la distance de Von mises
Ri=rang(BCP_R,BCP_R)
Si=rang(BOA_R,BOA_R)
n=length(BCP_R)
Deheuvels_cop=function(u,v){
  cont=0
  for (i in 1:n) {
    if((isTRUE(Ri[i]/n)<=u) & isTRUE(Si[i]/n<=v)){cont=cont+1}
  }
  return((1/n)*cont)
}
# Construction de la distance de von mise !
Distance_Cr_v_mise=function(N,copule,nu){

  if (copule=='student') {
    g=Student_cop
    som=0
    for (i in 1:N) {
```

```
      val=(g(u=Ri[i]/(n+1),v=Si[i]/(n+1),nu=nu)-Deheuvels_cop(u=Ri[i]/
(n+1),v=Si[i]/(n+1)))^2
      som=som+val
    }
    K=som
    return(K)
  }
  # Les autres copules
  if (copule=='Gumbel') g=Gumbel_cop
  if (copule=='Clayton') g=clayton_cop
  if (copule=='Frank')  g=Frank_cop
  som=0
  for (i in 1:N) {
    val=(g(Ri[i]/(n+1),Si[i]/(n+1))-Deheuvels_cop(Ri[i]/(n+1),Si[i]/
(n+1)))^2
    som=som+val
  }
  K=som
  return(K)
}
# Distance de kolmogorov smirnov
KS_dist=function(N,copule){
  v=vector(length = N)
  if (copule=='Gumbel') g=Gumbel_cop
  if (copule=='Clayton') g=clayton_cop
  if (copule=='Frank')  g=Frank_cop

  for (i in 1:N) {
    v[i]=abs(g(Ri[i]/(n+1),Si[i]/(n+1))-Deheuvels_cop(Ri[i]/(n+1),Si[i]/
(n+1)))
  }
  k=max(v)
  return(k)

}
# Distance de Von mise Cramér:
#Frank
l0=Distance_Cr_v_mise(N = 1,copule = "Frank",nu = 7)
l00=Distance_Cr_v_mise(N = 2,copule = "Frank",nu = 7)
# gumbel
l1=Distance_Cr_v_mise(N = 1,copule = "Gumbel",nu = 7)
l2=Distance_Cr_v_mise(N = 2,copule = "Gumbel",nu = 7)
# Clayton
l3=Distance_Cr_v_mise(N = 1,copule = "Clayton",nu = 7)
l4=Distance_Cr_v_mise(N = 2,copule = "Clayton",nu = 7)
# Student
l5=Distance_Cr_v_mise(N = 1,copule = "student",nu = 9)
l6=Distance_Cr_v_mise(N = 2,copule = "student",nu = 9)
# Distance de Kolmogorov Smirnov
# gumbel
l8=KS_dist(N = 2,copule = "Gumbel")
# Clayton
l9=KS_dist(N = 2,copule = "Clayton")
#frank
l10=KS_dist(N = 2,copule = "Frank")
#remplir les ditances dans un tableau
DV=data.frame(DISTANCE=0:2,Gumbel=0:2,Clayton=0:2,Frank=0:2,student=0:2,choix_optimal=0
DV$DISTANCE=c('D1','D2','D infini')
```

```
DV$Clayton=c(l1,l2,l8)
DV$Gumbel=c(l3,l4,l9)
DV$student=c(l5,l6,'-')
DV$Frank=c(l0,l00,l10)
DV$choix_optimal=c("Gumbel","Gumbel","Gumbel")
```