Simulación

Lab: Simulación y Estimación de cópulas en R

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Últimas dos semanas de clase diciembre de 2018

Simulación de cópulas bivariadas

Simulación de cópulas bivariadas

- El siguiente método para generar muestras de cópulas bivariadas fue propuesto por Nelsen (2006). La función de distribución condicional de V, dado $U = u$, se puede escribir como:

$$
\begin{aligned}
C_{V|U}(u,v) = P(V \leq v | U = u) &= \frac{C(u,v)}{C(u,1)} \\
&= \lim_{\epsilon \to 0} \frac{C(u+\epsilon, v) - C(u,v)}{\epsilon} \\
&= \frac{\partial C(u,v)}{\partial u}
\end{aligned}
$$

- Basado en lo anterior, un algoritmo general para extraer muestras de una cópula $C(u,v)$ es el siguiente:

Muestreo de observaciones de una cópula bivariada

1. Extrae dos variadas uniformes independientes u_1 y v_2.
2. Hacer $u_2 = C^{-1}_{V|U=u_1}(u_1, v_2)$
3. El vector (u_1, u_2) es el generado de la cópula C.

Ejemplo I

En muchas aplicaciones prácticas se utiliza la cópula general de Farlie-Gumbel-Morgensten (FMG):

$$C_\theta(u, v) = uv + \theta u(1-u)v(1-v)$$

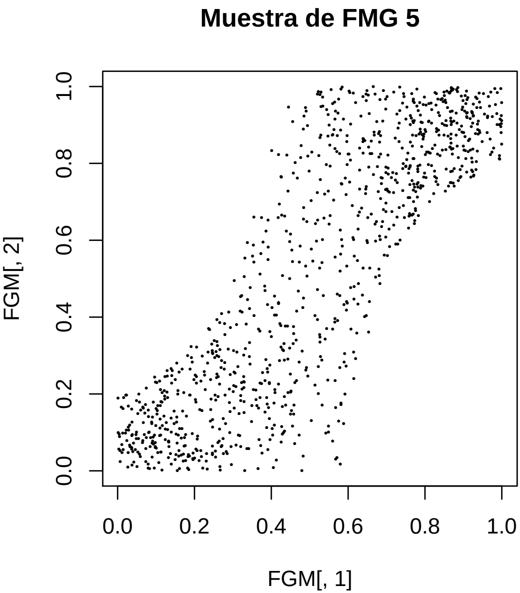
Evaluando $C_{V|U}(u, v) = v(1 + \theta(1 - 2u)) + \theta(1 - 2u)v$. Para esta cópula el algoritmo propuesto es de la forma siguiente (hay que hacer el ejercicio de inversión):

1. extrae dos uniformes independientes (u_1, v_2)
2. Define: $u_2 = \frac{2v_2}{A+B}$, donde $A = 1 + \theta(1 - 2u_1)$, $B = \sqrt{A^2 - 4(A-1)v_2}$

Por ejemplo, para $\theta = 5$:

```
theta <- 5
c1 <- cbind(runif(1000),runif(1000)) # genera u y v
A <- 1+theta*(1-2*c1[,1])
B <- sqrt(A^2-4*(A-1)*c1[,2])
FGM <- cbind(c1[,1],2*c1[,2]/(A+B))
par(pty="s"); plot(FGM[,1],FGM[,2],pch=16,cex=0.3,main=paste("Muestra de FMG",theta))
```

Ejemplo II



Paquete `copula`

Características del paquete `copula`

- El paquete `copula` desarrollada por Jun Yan (2006) provee una plataforma para la modelación multivariada de cópulas.
- Incluye, para las cópulas elípticas (normal, t), y arquimedianas (Clayton, Frank, Gumbel), los siguientes métodos
  - Evaluación de densidad/distribución
  - Generación de muestras de la cópula
  - Visualización
  - Ajuste de modelos basados en cópulas, utilizando máxima verosimilitud.
  - La cópula de valores extremos sólo está implementada para el caso bivariado.
- Otro paquete importante es el paquete `fCopulae` de Tobias Setz, como parte de los paquetes financieros de `Rmetrics`, que complementa los modelos de cópula para incluir las cópulas de Valor Extremo (se ve en la siguiente lámina), y la cópula empírica (con la definición usual de distribución empírica).

Sea $A:[0,1]\to[1/2,1]$ una función convexa que satisface la siguiente condición para $w\in[0,1]$: $\max\{w,1-w\}\leq A(x)\leq 1$. Las familia de cópulas de valores extremos (Pickands, 1981) se define a partir de la función A como:

$$C(u,v)=\exp\left[\log(uv)A\left(\frac{\log(v)}{\log(uv)}\right)\right]$$

Algunos ejemplos de casos particulares:

- $A(w)=1$ es la cópula de independencia.
- $A(w)=\max\{w,1-w\}$ es la cópula comonotónica.
- $A(w)=\left[w^\theta+(1-w)^\theta\right]^{1/\theta}$, $\theta\geq 1$ es la cópula de Gumbel.
- $A(w)=1-\left[w^{-\theta}+(1-w)^{-\theta}\right]^{-1/\theta}$, $\theta\geq 0$ es la cópula de Galambos.

Hay muchísimas otras que se han desarrollado para aplicaciones en seguros y finanzas.

Clases definidas en el paquete `copula` I

Hay básicamente dos clases de objetos:

`copula` para definir cópulas:

$$C(u_1, \ldots, u_p) = F(F_1^{-1}(u_1), \ldots, F_p^{-1}(u_p))$$

donde las funciones F y F_i son dadas.

`mvdc` para definir distribuciones multivariadas a partir de cópulas:

$$F(x_1, \ldots, x_p) = C(F_1(x_1), \ldots, F_p(x_p))$$

donde C y F_i son dadas.

Clase `copula` I

La clase `copula` considera las subclases:

- `ellipCopula`, incluye `normalCopula` y `tCopula`. Como se vió en clase, basta con definir la matriz de correlaciones como parámetro de las dos familias, y en el caso de la t también se requieren los grados de libertad (`df`).
- La matriz de correlaciones define la estructura de dependencia, y se pueden usar las siguientes configuraciones:

  `ar1`: especificación para dependencias autorregresivas, por ejemplo, con $p = 3$:
  $$\begin{pmatrix} 1 & \rho_1 & \rho_1^2 \\ \rho_1 & 1 & \rho_1 \\ \rho_1^2 & \rho_1 & 1 \end{pmatrix}$$

  `ex`: exchangeable: todas las variables tienen la misma correlación
  $$\begin{pmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_1 \\ \rho_1 & \rho_1 & 1 \end{pmatrix}$$

`toep` Toeplitz: matriz con estructuras diagonales constantes:

$$\begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{pmatrix}$$

`un` unstructured: todas las correlaciones son diferentes.

$$\begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_3 \\ \rho_2 & \rho_3 & 1 \end{pmatrix}$$

- `archmCopula`, que incluye `claytonCopula`, `frankCopula` y `gumbelCopula`.

Ejemplo 1

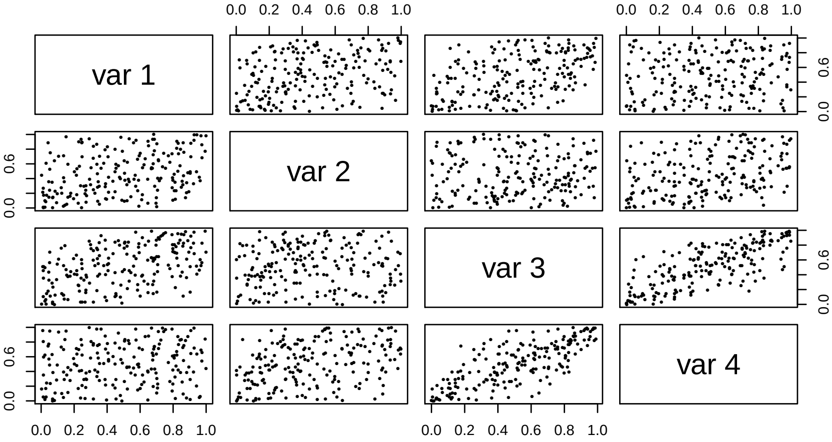
Para generar una cópula gaussiana de dimensión 4 de tipo no estructurado, se requiere definir 6 valores de correlaciones:

```
library(copula)
copula.normal4 <- ellipCopula(family = "normal", dim = 4, dispstr = "un",
                  param = c(0.4,0.5,0.2,0,0.3,0.8))
copula.normal4 #objeto de clase normalCopula

Normal copula, dim. d = 4
Dimension:  4
Parameters:
  rho.1   = 0.4
  rho.2   = 0.5
  rho.3   = 0.2
  rho.4   = 0.0
  rho.5   = 0.3
  rho.6   = 0.8
dispstr:  un

u <- rCopula(200,copula.normal4) #genera observaciones de la cópula construida
cor(u)

          [,1]       [,2]       [,3]      [,4]
[1,] 1.0000000 0.32230365 0.47229092 0.1323811
[2,] 0.3223036 1.00000000 0.04323918 0.3643341
[3,] 0.4722909 0.04323918 1.00000000 0.7616331
[4,] 0.1323811 0.36433410 0.76163306 1.0000000
```

Una gráfica de la función anterior en pares:

Clase copula IV

```
pairs(u,pch=16, cex=0.5)
```



Ejemplo 2

Para generar una cópula t de dimensión 3 de tipo Toeplitz, se requiere definir 2 valores de correlaciones y los grados de libertad:

```
micopula.t3 <- ellipCopula(family = "t", dim = 3, dispstr = "toep",
param = c(0.8,0.5), df = 8)
micopula.t3 #objeto de clase tCopula

t-copula, dim. d = 3
Dimension:  3
Parameters:
  rho.1    = 0.8
  rho.2    = 0.5
  df       = 8.0
dispstr:  toep

rCopula(5,micopula.t3) #genera cinco observaciones de la cópula construida

          [,1]      [,2]      [,3]
[1,] 0.9875935 0.9561816 0.9725253
[2,] 0.3291993 0.2426363 0.1072573
[3,] 0.9674859 0.9878801 0.9623460
[4,] 0.5196940 0.4088097 0.1943940
[5,] 0.1980348 0.1928345 0.1462435
```

Ejemplo 3

Para generar una cópula tipo Clayton bidimensional con parámetro $\theta = 2$:

```
clayton2 <- archmCopula(family = "clayton", dim = 2, param = 2)
clayton2 #el programa llama alpha al parámetro

Clayton copula, dim. d = 2
Dimension:  2
Parameters:
  alpha    = 2

# Generemos una muestra de ésta cópula:
y <- rCopula(1000,clayton2)
```

Para comparar con las curvas de nivel de la cópula de Clayton, podemos generar una muestra aleatoria y ver su distribución conjunta. Notemos que la cópula de Clayton tiene una alta de concentración de probabilidad en el origen. Esto puede ser de utilidad para correlacionar pequeñas pérdidas.

```
par(mfrow=c(1,2))
contour(clayton2,dCopula) #gráfica de curvas de nivel
plot(y,cex=0.3)
```

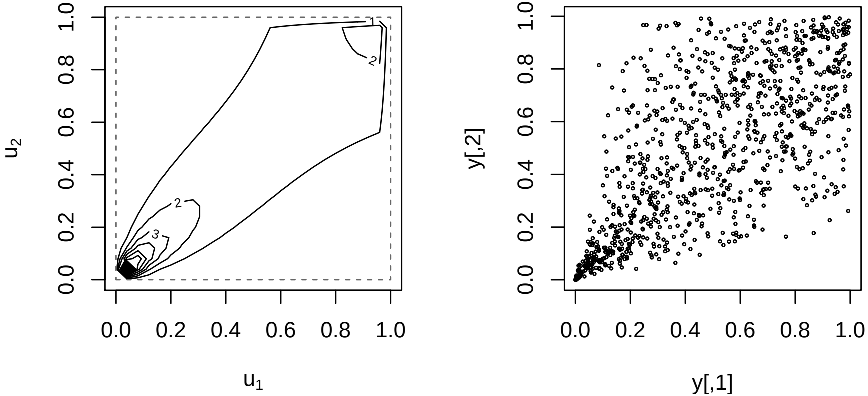

1.0
0.8
0.6
0.4
0.2
0.0
u_2
u_1
1
2
2
3
y[,2]
y[,1]

Tabla de cópulas Arquimedianas I

Familia	Espacio parametral θ	$\psi(t)$	$\psi^{-1}(t)$	$C(u,v)$
Clayton	$\theta \geq 0$	$\frac{t^{-\theta}-1}{\theta}$	$(1+\theta t)^{-1/\theta}$	$(u^{-\theta}+v^{-\theta}-1)^{-1/\theta}$
Frank	$\theta \geq 0$	$-\log \frac{e^{-\theta t}-1}{e^{-\theta}-1}$	$-\theta^{-1}\log(1+e^{-t}(e^{-\theta}-1))$	$-\frac{1}{\theta}\log\left(1+\frac{(e^{-\theta u}-1)e^{-\theta v}-1)}{e^{-\theta}-1}\right)$
Gumbel	$\theta \geq 1$	$(-\log t)^{\theta}$	$e^{-t^{1/\theta}}$	$\exp\left[-((-\log u)^{\theta}+(-\log v)^{\theta})^{1/\theta}\right]$

Clase `mvdc`

- Esta clase de objetos está diseñada para construir distribuciones multivariadas con maginales dadas usando cópulas.
- Este es el caso que hicimos en la última clase de cópulas, dadas las marginales y una estructura de dependencia, construir una distribución conjunta usando, por ejemplo, la cópula Gaussiana.
- Esta clase tiene tres componentes:

  `copula`: Especifica la cópula C a usar para 'pegar' las marginales.

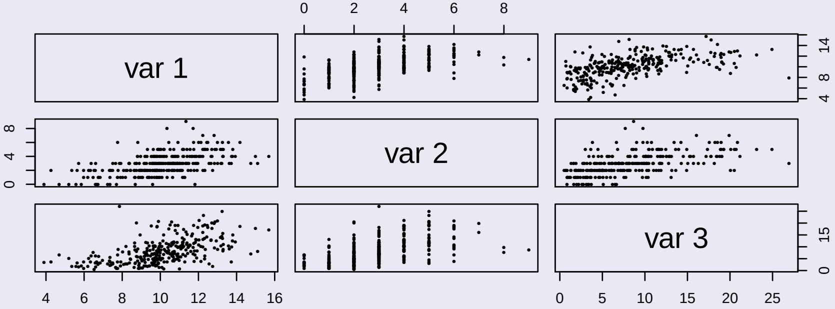
  `margins`: Especifica los nombres de las distribuciones marginales a usar.

  `paramMargins`: una lista de listas, con los parámetros de las distribuciones marginales.

Ejemplo 4

Generar una distribución conjunta con una cópula Frank con parámetro $\theta = 5$ y con marginales $\mathcal{N}(10,4)$, $\mathcal{P}(3)$ y $\mathcal{G}(2,4)$.

```
copula.Frank5 <- archmCopula(family = "frank", dim = 3, param = 5)
micopula <- mvdc(copula = copula.Frank5, margins = c("norm","pois","gamma"),
                 paramMargins = list(list(mean=10,sd=2), list(lambda=3), list(shape=2,scale=4)))
u <- rMvdc(300,micopula) #muestra aleatoria
par(mar=c(1,1,1,1));pairs(u,pch=16,cex=0.5)
```



Funciones de distribución y densidad para cópulas I

Para la distribución conjunta creada a partir de la cópula

```
u <- rMvdc(5,micopula)
u # puntos del dominio

          [,1] [,2]      [,3]
[1,] 11.676336    3  8.375328
[2,]  9.249461    0  3.149692
[3,]  6.258955    0  8.201856
[4,] 10.448339    2 10.317386
[5,]  9.309328    5  5.082003

dMvdc(u,micopula) # puntos de la densidad

[1] 0.0048161699 0.0017475568 0.0001045695 0.0029185246 0.0007598814

pMvdc(u,micopula) # puntos de la distribución

[1] 0.487273665 0.023852428 0.006146711 0.345627418 0.243610613
```

Para la cópula dada:

Funciones de distribución y densidad para cópulas II

```r
u <- rCopula(5,copula.Frank5)
u # puntos del dominio
```

```
          [,1]      [,2]      [,3]
[1,] 0.1602444 0.6381590 0.2802389
[2,] 0.5283327 0.3424339 0.1241710
[3,] 0.1103816 0.5019964 0.5866382
[4,] 0.5176543 0.7096821 0.5726154
[5,] 0.6738563 0.6959941 0.8408561
```

```r
dCopula(u,copula.Frank5) # puntos de la densidad
```

```
[1] 0.7632402 0.8930690 0.3534685 2.0514131 2.7506462
```

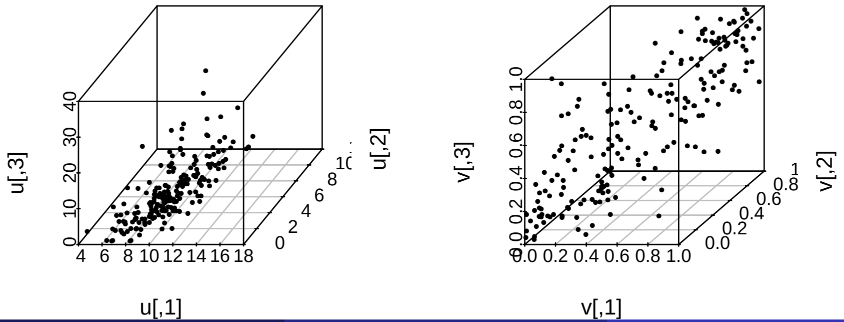
```r
pCopula(u,copula.Frank5) # puntos de la distribución
```

```
[1] 0.10343064 0.08826976 0.09366839 0.39107513 0.54487777
```

Gráficas

La siguiente gráfica muestra la realización de una muestra de la distribución conjunta solicitada:

```r
library(scatterplot3d)
par(mfrow=c(1,2),mar=c(1,2,1,1),oma=c(0,0,1,1),mgp=c(2,1,0))
u <- rMvdc(200,micopula)
scatterplot3d(u,cex.symbols=0.5,pch=16)
v <- rCopula(200,copula.Frank5)
scatterplot3d(v,cex.symbols=0.5,pch=16)
```



Contornos de funciones de densidad/distribución I

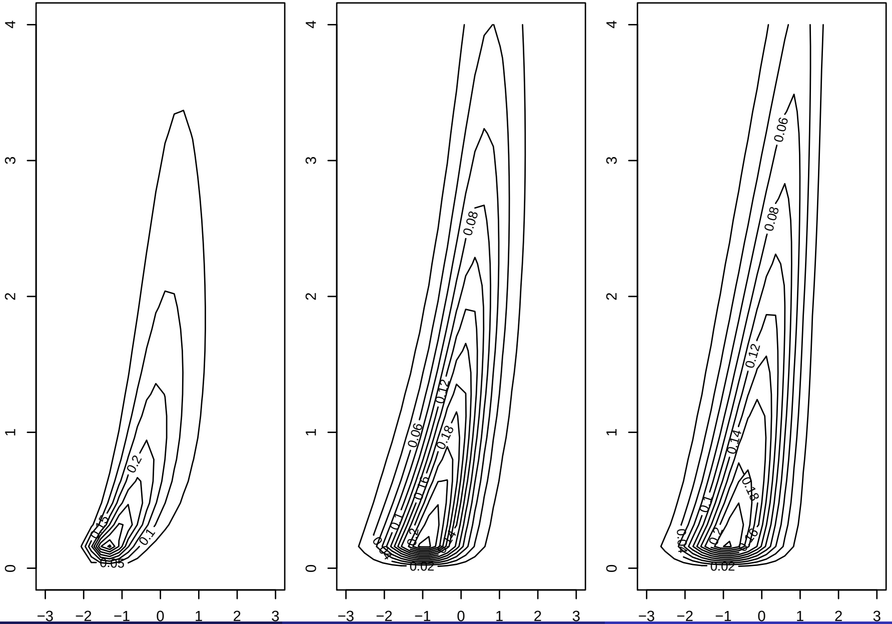
El siguiente código grafica los contornos de densidad de distribuciones bivariadas definidas con las tres cópulas de Clayton, Frank y Gumbel con marginales normales.

```
miMvd1 <- mvdc(copula = archmCopula(family="clayton", param=2), margins = c("norm", "gamma"),
               paramMargins = list(list(mean=0,sd=1), list(shape=1,scale=2)))
miMvd2 <- mvdc(copula = archmCopula(family="frank", param=5.763), margins = c("norm", "gamma"),
                           paramMargins = list(list(mean=0,sd=1), list(shape=1,scale=2)))
miMvd3 <- mvdc(copula = archmCopula(family="gumbel", param=2), margins = c("norm", "gamma"),
                           paramMargins = list(list(mean=0,sd=1), list(shape=1,scale=2)))
```

Los parámetros han sido escogidos para dar una τ de Kendall para las tres distribuciones igual a 0.5.

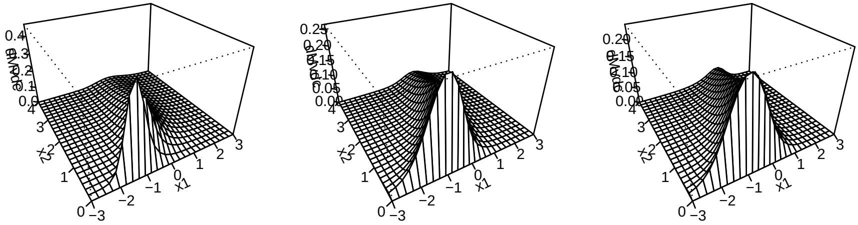
```
par(mfrow=c(1,3), mar=c(2,2,1,1), oma=c(1,1,0,0), mgp=c(2,1,0))
contour(miMvd1, dMvdc,xlim=c(-3,3), ylim=c(0,4))
contour(miMvd2, dMvdc,xlim=c(-3,3), ylim=c(0,4))
contour(miMvd3, dMvdc,xlim=c(-3,3), ylim=c(0,4))
```

Contornos de funciones de densidad/distribución II



La función `persp` es similar:

```
par(mfrow=c(1,3), mar=c(2,2,1,1), oma=c(1,1,0,0), mgp=c(2,1,0))
persp(miMvd1, dMvdc,xlim=c(-3,3), ylim=c(0,4))
persp(miMvd2, dMvdc,xlim=c(-3,3), ylim=c(0,4))
persp(miMvd3, dMvdc,xlim=c(-3,3), ylim=c(0,4))
```



Estimación de cópulas

- Un problema importante es: dado un conjunto de datos, elegir una cópula para ajustarlos.
- El problema tiene dificultades técnicas y trampas con las que hay que ser muy cuidadoso.
- El problema es que la estimación de cópulas implica usualmente que cada marginal debe ser evaluada y conectada a una distribución multivariada estimada.
- A continuación veremos un ejemplo de estimación.
- Usaremos varios paquetes de `R` para realizar el ejercicio:
  - paquetes `Ecdat` para tomar algunos datos
  - `copula`, que es el paquete principal de donde se obtienen la mayoría de las características.
  - `fGarch` para el uso de la densidad t estandarizada
  - `MASS` para el uso de las funciones *fitdistr, kde2d*
  - `fCopulae` para funciones adicionales de copulas: *pempiricalCopula, ellipticalCopulafit.*

Datos I

- El paquete `Ecdat` es un conjunto de datos Econométricos. Los datos `CRSPday` contiene una base de datos de rendimientos diarios de acciones del Center for Research in Security Prices (CRSP), del 3 de enero de 1969 al 31 de diciembre de 1998.
- Nos vamos a fijar en las dos variables `ibm`, que es el rendimiento de IBM y `crsp` que es un índice ponderado de rendimientos construído por el CRSP.

Datos II

```r
suppressMessages(library(Ecdat)) # fuente de datos
library(copula)
suppressMessages(library(fGarch)) # función de densidad t estandarizada
suppressMessages(library(MASS)) # usa las funciones fitdistr y kde2d
suppressMessages(library(fCopulae))  # funciones adicionales de copula (pempiricalCopula y ellipticalCopulaFit)

data(CRSPday,package="Ecdat")
head(as.data.frame(CRSPday)) # muestra la estructura de los datos
```

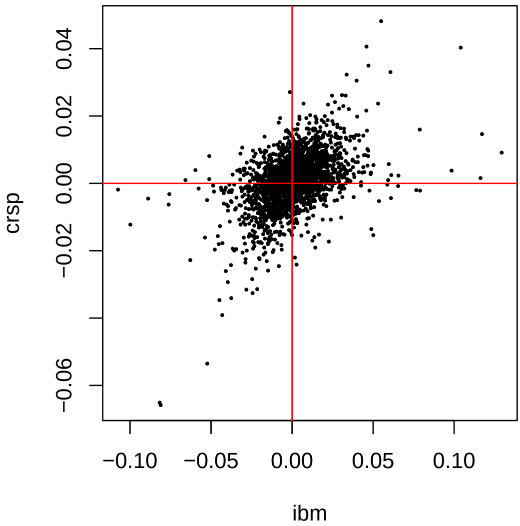
```
  year month day        ge       ibm     mobil      crsp
1 1989     1   3 -0.016760  0.000000 -0.002747 -0.007619
2 1989     1   4  0.017045  0.005128  0.005510  0.013016
3 1989     1   5 -0.002793 -0.002041  0.005479  0.002815
4 1989     1   6  0.000000 -0.006135  0.002725  0.003064
5 1989     1   9  0.000000  0.004115  0.005435  0.001633
6 1989     1  10 -0.005602 -0.007172  0.008108 -0.001991
```

```r
ibm <- CRSPday[,5]; crsp <- CRSPday[,7]
n <- length(ibm); n #número de observaciones
```

```
[1] 2528
```

```r
par(pty = "s"); plot(ibm, crsp, cex = 0.4, pch = 16)
abline(h = 0, v = 0, col="red")
```

Datos III



Marginales propuestas

- A continuación se ajustará una distribución t a cada una de las variables marginales. Los valores que se guardan corresponden a los valores estimados de las distribuciones t marginales. Cada distribución marginal puede ajustar diferentes grados de libertad.
- La función `fitdistr` del paquete `MASS` estima las características de una función de distribución usando máxima verosimilitud (en el caso de la t, su media, escala, y grados de libertad).

```
est.ibm <- as.numeric(fitdistr(ibm,"t")$estimate) #parámetros t: media, escala, gl
est.crsp <- as.numeric(fitdistr(crsp,"t")$estimate)
        #Convierte los parámetros de escala a desviaciones estándar en el caso de la t
est.ibm[2] <- est.ibm[2]*sqrt(est.ibm[3]/(est.ibm[3]-2))
est.crsp[2] <- est.crsp[2]*sqrt(est.crsp[3]/(est.crsp[3]-2))
        #Grados de libertad para cada caso
est.ibm[3]

[1] 4.276156

est.crsp[3]

[1] 3.473982
```

Ajuste de cópula específica I

- Como un ejercicio inicial, supongamos que se quiere ajustar una cópula específica, por ejemplo, una cópula t.
- Para estimar una t-cópula por máxima verosimilitud, se requiere una estimación de la correlación y un valor inicial adecuado.
- Se usarán las densidades t estimadas como valores iniciales. Se define la cópula t con 2 grados de libertad

```
tau <- cor(ibm,crsp,method = "kendall")
omega <- 2/pi*asin(tau)
c(tau,omega)

[1] 0.3308049 0.2146404

copula2 <- tCopula(omega,dim=2,dispstr = "un",df = 2)
```

Ahora hay que ajustar la copula a los datos uniformes transformados:

Ajuste de cópula específica II

```
# La función pstd es la distribución estándar t
# por el método de máxima verosimilitud
d1 <- cbind(pstd(ibm, mean = est.ibm[1], sd = est.ibm[2], nu = est.ibm[3]),
                   pstd(crsp, mean = est.crsp[1], sd = est.crsp[2], nu = est.crsp[3]))

fit1 <- fitCopula(copula2, method = "ml", optim.method = "L-BFGS-B", data = d1,
                                  start = c(omega,5), lower = c(0,2.5), upper = c(0.5,15))
fit1

Call: fitCopula(copula, data = data, method = "ml", start = ..3, lower = ..4,
    upper = ..5, optim.method = "L-BFGS-B")
Fit based on "maximum likelihood" and 2528 2-dimensional observations.
Copula: tCopula
 rho.1     df
0.4937 9.8537
The maximized loglikelihood is 362
Optimization converged
```

Cópulas alternas I

Para efectos de comparación, consideremos ahora el ajuste de otras cópulas a los datos:

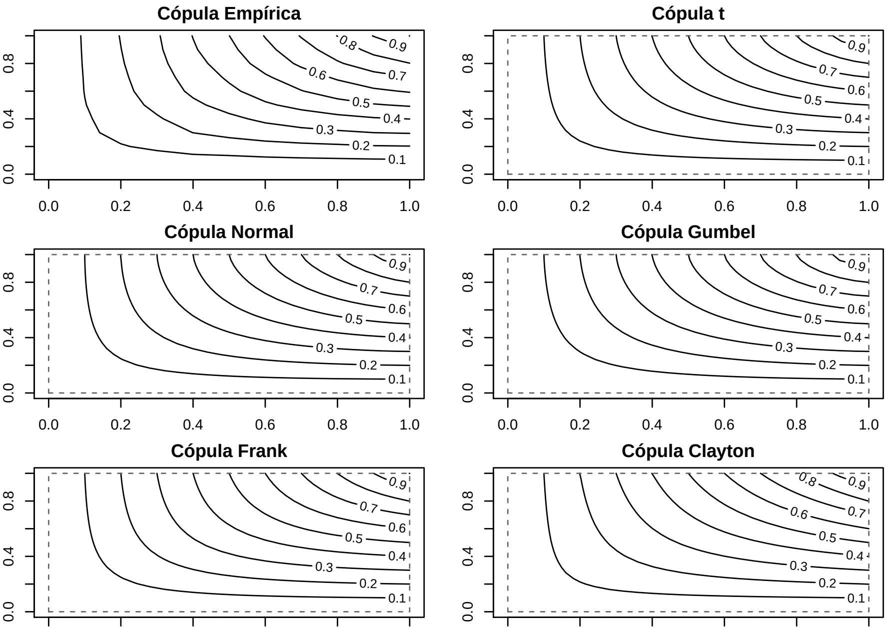
```
#Ajusta copula normal
        fnorm <- fitCopula(data=d1,copula=normalCopula(-0.3,dim=2),
        method="ml",optim.method="BFGS",start=0.5)
#Ajusta Gumbel
        fgumbel <- fitCopula(data=d1,copula=gumbelCopula(3,dim=2),
        method="ml",optim.method="BFGS",start=2)
#Ajusta Frank
        ffrank <- fitCopula(data=d1,copula=frankCopula(3,dim=2),
        method="ml",optim.method="BFGS",start=2)
#Ajusta Clayton
        fclayton <- fitCopula(data=d1,copula=claytonCopula(3,dim=2),
        method="ml",optim.method="BFGS",start=2)
```

Las copulas estimadas se compararán contra la cópula empírica y se verá si hay alguna estimación que quede cerca a la cópula que se obtiene de los datos.

Comparación gráfica I

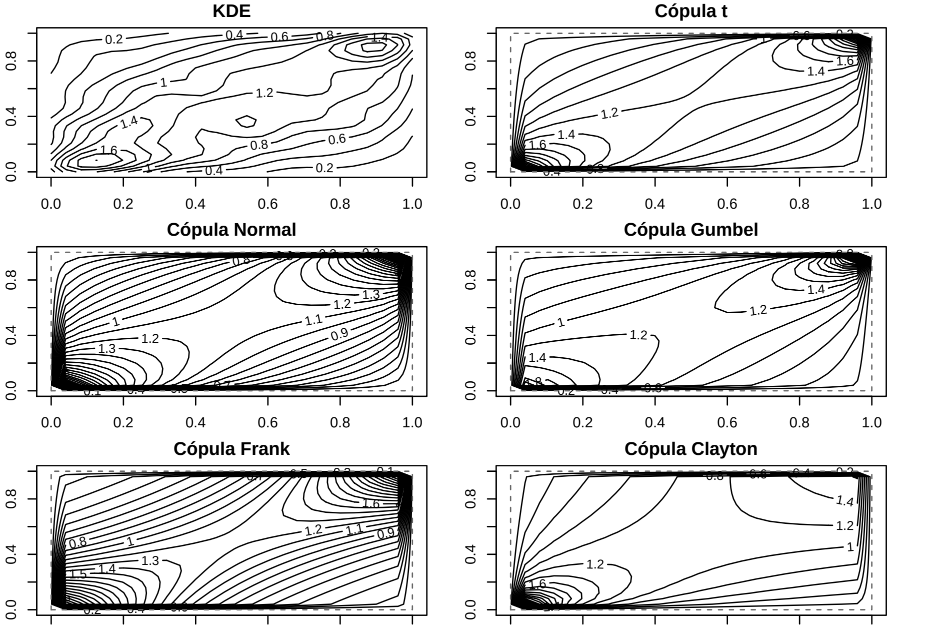
```r
u <- d1
dem <- pempiricalCopula(u[,1],u[,2])
par(mfrow=c(3,2),mar=c(2,2,2,2))
contour(dem$x,dem$y,dem$z,main="Cópula Empírica")
contour(tCopula(fit1@estimate[1],df=round(fit1@estimate[2],0)),pCopula,main="Cópula t")
contour(normalCopula(fnorm@estimate),pCopula,main="Cópula Normal")
contour(gumbelCopula(fgumbel@estimate),pCopula,main="Cópula Gumbel")
contour(frankCopula(ffrank@estimate),pCopula,main="Cópula Frank")
contour(claytonCopula(fclayton@estimate),pCopula,main="Cópula Clayton")
```

Comparación gráfica II



En el siguiente conjunto de gráficas se comparará la estimación de la densidad bivariada entre las diferentes estimaciones paramétricas

```
par(mfrow=c(3,2),mar=c(2,2,2,2))
contour(kde2d(u[,1],u[,2]),main="KDE")
contour(tCopula(fit1@estimate[1],df=fit1@estimate[2]),dCopula,
main="Cópula t",nlevels=25)
contour(normalCopula(fnorm@estimate),dCopula,main="Cópula Normal",nlevels=25)
contour(gumbelCopula(fgumbel@estimate),dCopula,main="Cópula Gumbel",nlevels=25)
contour(frankCopula(ffrank@estimate),dCopula,main="Cópula Frank",nlevels=25)
contour(claytonCopula(fclayton@estimate),dCopula,main="Cópula Clayton",nlevels=25)
```


KDE
Cópula t
Cópula Normal
Cópula Gumbel
Cópula Frank
Cópula Clayton
0.0
0.2
0.4
0.6
0.8
1.0

Evaluación a través de AIC

Por último, podemos comparar los AIC. Recuerden que el criterio de información de Akaike se utiliza para comparar modelos. Aquel con el mayor valor absoluto nos dice cuál es el mejor modelo:

```
#AIC Normal
2*length(fnorm@estimate)-2*fnorm@loglik

[1] -692.3688

#AIC Gumbel
2*length(fgumbel@estimate)-2*fgumbel@loglik

[1] -624.4514

#AIC frank
2*length(ffrank@estimate)-2*ffrank@loglik

[1] -648.5734

#AIC Clayton
2*length(fclayton@estimate)-2*fclayton@loglik

[1] -584.2204

#AIC t
2*length(fit1@estimate)-2*fit1@loglik

[1] -719.9693
```