

Simulación

Estimación de cópulas

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Clase Optativa Sábado 14, Abril 2018

Características del paquete

- 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 números aleatorios 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, y la cópula Empírica.

Clases definidas en el paquete `copula` I

Hay básicamente dos clases:

`copula` para definir cópulas:

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

donde F y F_i son dadas.

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

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

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 case 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

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

Clase copula II

`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`.

Clase copula III

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 cuatro observaciones de la cópula construida
cor(u)
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1.00000000	0.389097211	0.355995209	0.07384396
[2,]	0.38909721	1.000000000	-0.007264663	0.24271108
[3,]	0.35599521	-0.007264663	1.000000000	0.81953191
[4,]	0.07384396	0.242711076	0.819531911	1.00000000

Clase copula IV

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 cuatro observaciones de la cópula construida
```

	[,1]	[,2]	[,3]
[1,]	0.04891966	0.03781008	0.08363567
[2,]	0.99667312	0.98516667	0.94852586
[3,]	0.63299559	0.56887747	0.43112456
[4,]	0.34509851	0.53663246	0.77997529
[5,]	0.70787534	0.57012189	0.78469301

Clase copula V

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
```

```
par(mar=c(0,0,0,0))
contour(clayton2,dCopula) #gráfica de curvas de nivel
```

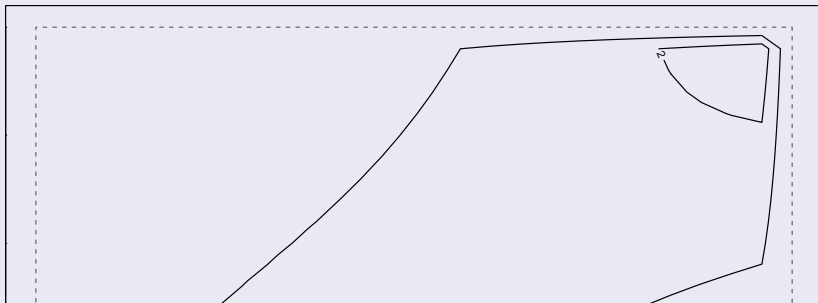


Tabla de cópulas Arquimedianas I

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 I

Esta clase está diseñada para construir distribuciones multivariadas con marginales 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 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.

Ejemplos `mvdc` I

Ejemplos mvdc II

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,] 10.110978    6  7.541412
[2,] 11.070421    5 15.026502
[3,]  9.500377    2 14.408854
[4,]  8.601204    3  4.442726
[5,] 12.098194    4  7.295075

dMvdc(u,micopula) # puntos de la densidad

[1] 0.0006243805 0.0012417815 0.0007189697 0.0031524325 0.0019509686

pMvdc(u,micopula) # puntos de la distribución

[1] 0.4168085 0.6543372 0.2870078 0.1530373 0.4996093
```

Para la cópula dada:

Funciones de distribución y densidad para cópulas II

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

      [,1]      [,2]      [,3]
[1,] 0.1769304 0.29319620 0.3913960
[2,] 0.5733843 0.09463942 0.2104247
[3,] 0.4904138 0.82311019 0.3087709
[4,] 0.4168110 0.55471102 0.4454344
[5,] 0.7297805 0.76328456 0.5594176

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

[1] 2.1262221 0.8630446 0.5735764 2.1627117 2.0727310

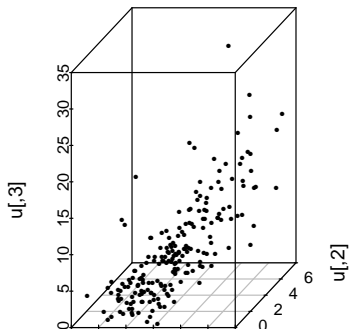
pCopula(u, copula.Frank5) # puntos de la distribución

[1] 0.09986995 0.05345691 0.25219056 0.27123403 0.47423642
```

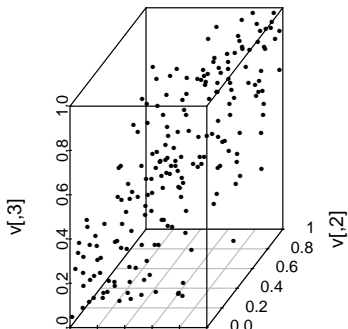
Gráficas

La siguiente gráfica muestra la realización de una muestra

```
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)
```



Jorge de la Vega Góngora



Simulación

Contornos de funciones de densidad/distribución I

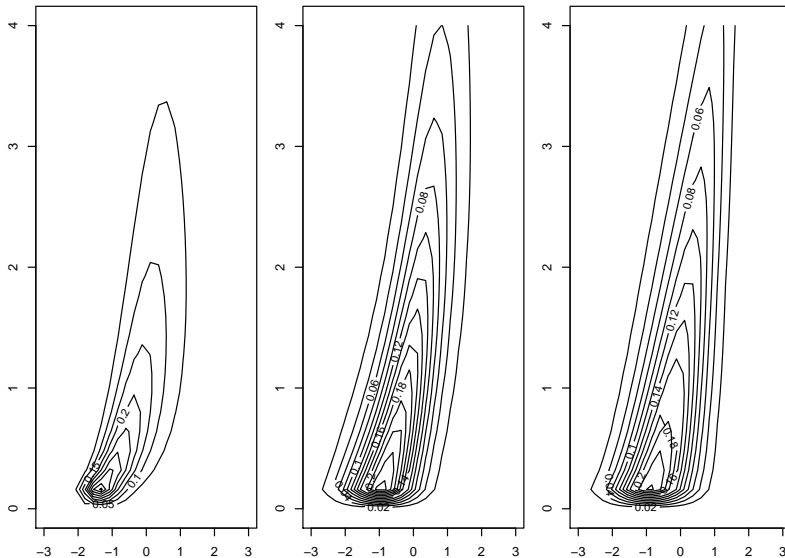
El siguiente código grafica los contornos de densidad de distribuciones bivariadas definidas con las tres cópulas de Clayton, Frank y Gumbel con marginales normales.

```
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

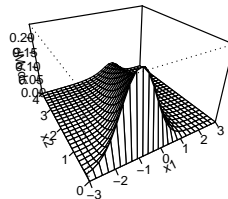
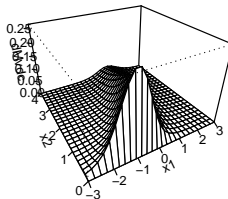
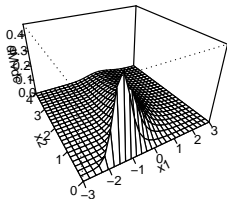


```
par(mfrow=c(1,3), mar=c(2,2,1,1), oma=c(1,1,0,0), mgp=c(2,1,0))
```

persp

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))
```



2.8 Estimación de cópulas

Introducción

- Dado un conjunto de datos, elegir una cópula para ajustarlos en un problema importante pero difícil y llena de trucos. 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. En particular se usarán los 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* y *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 construido por el CRSP.

Datos II

```

suppressMessages(library(Ecdat)) #fuente de datos
library(copula)
library(fGarch) #función de densidad t estandarizada

Loading required package: timeDate
Loading required package: methods
Loading required package: timeSeries
Loading required package: fBasics

suppressMessages(library(MASS)) #usa las funciones fitdistr y kde2d
library(fCopulae) #funciones adicionales de copula (pempiricalCopula y ellipticalCopulaFit)

Loading required package: fMultivar

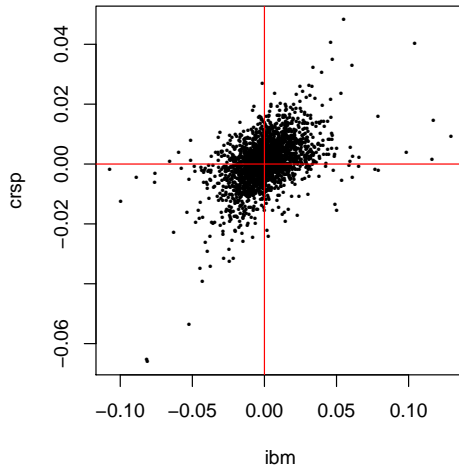
data(CRSPday, package="Ecdat")
ibm <- CRSPday[,5]; crsp <- CRSPday[,7]
n <- length(ibm); n #número de observaciones

[1] 2528

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)

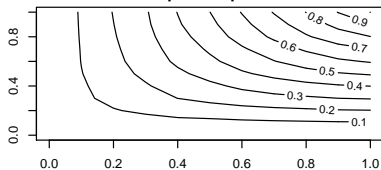
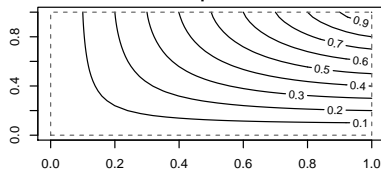
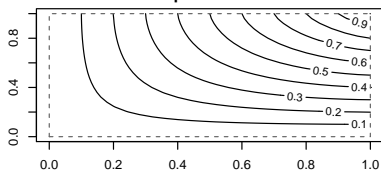
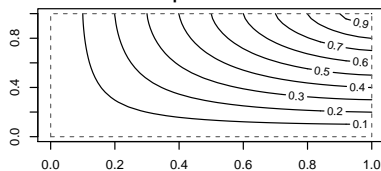
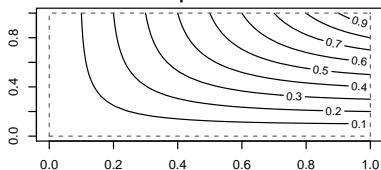
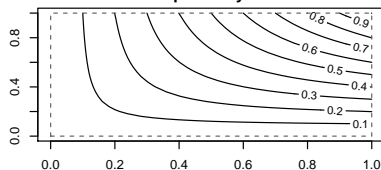
#Ajusta Clayton
fclayton <- fitCopula(data=d1, copula=claytonCopula(3, dim=2),
  method="ml", optim.method="BFGS", start=2)
```

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

```
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

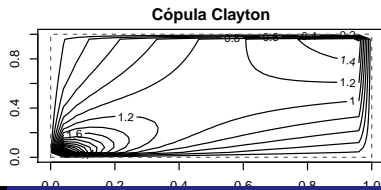
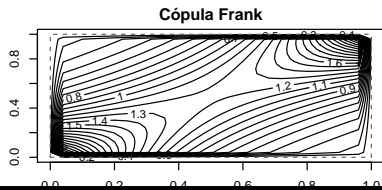
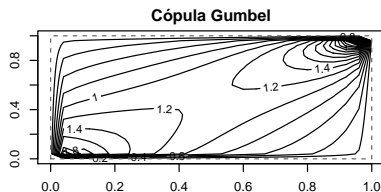
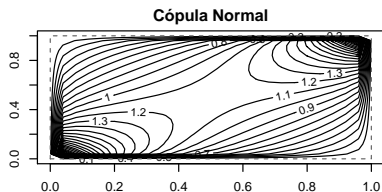
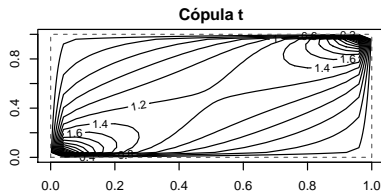
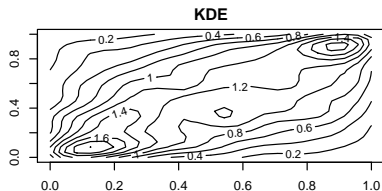
Cópula Empírica**Cópula t****Cópula Normal****Cópula Gumbel****Cópula Frank****Cópula Clayton**

Comparación de estimación bivariada con distribuciones paramétricas I

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)
```

Comparación de estimación bivariada con distribuciones paramétricas II



Evaluación a través de AIC

Por último, podemos comparar los AIC. Recuerden que $AIC =$

```
#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
```