Práctica de Detección de Anomalías

Laura Rodriguez Navas

20/07/2020

!MasterIA_GuionPracticas_Outliers_B1_1Variate_IQR

```
#############################################################################
# UNIVARIATE STATISTICAL OUTLIERS -> IQR
#############################################################################

# Siga las instrucciones indicadas en el fichero INSTRUCCIONES.txt.

# Vamos a trabajar con los siguientes objetos:

# mydata.numeric: frame de datos.
# indice.columna: Índice de una columna de datos de mydata.numeric.
# nombre.mydata:  Nombre del frame para que aparezca en los plots.

# En este script los estableceremos a la base de datos mtcars,
# columna 1 y nombre "mtcars"

mydata.numeric = mtcars[, -c(8:11)]  # mtcars[1:7]
indice.columna = 1
nombre.mydata = "mtcars"

# Ahora creamos los siguientes objetos:

# mydata.numeric.scaled -> Debe contener los valores normalizados demydata.numeric.
# Para ello, usad la función scale.
# columna -> Contendrá la columna de datos correspondiente a indice.columna.
# Basta realizar una selección con corchetes de mydata.numeric.
# nombre.columna -> Debe contener el nombre de la columna.
# Para ello, aplicamos la función names sobre mydata.numeric.
# columna.scaled -> Debe contener los valores normalizados de la anterior.

mydata.numeric.scaled = scale(mydata.numeric)
columna = mydata.numeric[, indice.columna]
nombre.columna = names(mydata.numeric)[indice.columna]
columna.scaled = mydata.numeric.scaled[, indice.columna]

#############################################################################
#############################################################################
# Parte primera. Cómputo de los outliers IQR
#############################################################################
#############################################################################
```

```r
#############################################################################
# Calcular los outliers según la regla IQR
# Directamente sin funciones propias
#############################################################################

# Transparencia 82

# Calculamos las siguientes variables:

# cuartil.primero -> primer cuartil
# cuartil.tercero -> tercer cuartil
# iqr -> distancia IQR

# Para ello, usamos las siguientes funciones:
# quantile(columna, x) para obtener los cuartiles
#    x=0.25 para el primer cuartil, 0.5 para la mediana y 0.75 para el tercero
# IQR para obtener la distancia intercuartil
#    (o bien reste directamente el cuartil tercero y el primero)

# Calculamos las siguientes variables -los extremos que delimitan los outliers-

# extremo.superior.outlier.normal  = cuartil tercero + 1.5 IQR
# extremo.inferior.outlier.normal  = cuartil primero - 1.5 IQR
# extremo.superior.outlier.extremo = cuartil tercero + 3 IQR
# extremo.inferior.outlier.extremo = cuartil primero - 3 IQR

# Construimos sendos vectores:

# vector.es.outlier.normal
# vector.es.outlier.extremo

# Son vectores de valores lógicos TRUE/FALSE que nos dicen
# si cada registro es o no un outlier con respecto a la columna fijada.
# Para ello, basta comparar con el operador > o el operador < la columna
# con alguno de los valores extremos anteriores.

# El resultado debe ser el siguiente:
# [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
# [18] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

# COMPLETAR

cuartil.primero <- quantile(columna, 0.25)
cuartil.tercero <- quantile(columna, 0.75)
iqr <- IQR(columna)

cuartil.primero
```

```
##    25%
## 15.425
```

```r
cuartil.tercero
```

```
##  75%
## 22.8
```

```r
iqr
```

```
## [1] 7.375
```

```r
extremo.superior.outlier.normal = cuartil.tercero + 1.5 * iqr
extremo.inferior.outlier.normal = cuartil.primero - 1.5 * iqr
extremo.superior.outlier.extremo = cuartil.tercero + 3 * iqr
extremo.inferior.outlier.extremo = cuartil.primero - 3 * iqr

extremo.superior.outlier.normal
```

```
##     75%
## 33.8625
```

```r
extremo.inferior.outlier.normal
```

```
##    25%
## 4.3625
```

```r
extremo.superior.outlier.extremo
```

```
##    75%
## 44.925
```

```r
extremo.inferior.outlier.extremo
```

```
##  25%
## -6.7
```

```r
vector.es.outlier.normal = columna > extremo.superior.outlier.normal |
  columna < extremo.inferior.outlier.normal

vector.es.outlier.extremo = columna > extremo.superior.outlier.extremo |
  columna < extremo.inferior.outlier.extremo

vector.es.outlier.normal
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
vector.es.outlier.extremo
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
##############################################################################
# Índices y valores de los outliers
##############################################################################

# Construimos las siguientes variables:

# claves.outliers.normales -> Vector con las claves (identificador numérico de fila) de
# los valores que son outliers. Para obtenerlo, usad which sobre vector.es.outlier.normal
# data.frame.outliers.normales -> data frame obtenido con la selección del data frame
# original de las filas que son outliers.
# Puede usarse o bien vector.es.outlier.normal o bien claves.outliers.normales.
# Este dataframe contiene los datos de todas las columnas de aquellas filas que son
# outliers.
# nombres.outliers.normales -> vector con los nombres de fila de los outliers.
# Para obtenerlo, usad row.names sobre el data frame anterior.
# valores.outliers.normales -> vector con los datos de los outliers. Se muestra sólo el
# valor de la columna que se fijó al inicio del script.
# Idem con los extremos

# Aplicando la selección dada por vector.es.outlier.normal:

#    [1] 20
#                    mpg cyl disp hp drat    wt qsec
#    Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9
#    [1] "Toyota Corolla"
#    [1] 33.9

# Aplicando la selección dada por vector.es.outlier.extremo:
# Ninguno

# COMPLETAR

claves.outliers.normales <- which(vector.es.outlier.normal == TRUE)
data.frame.outliers.normales <- as.data.frame(mydata.numeric[claves.outliers.normales, ])
nombres.outliers.normales <- row.names(mydata.numeric)[vector.es.outlier.normal == TRUE]
valores.outliers.normales <- columna[vector.es.outlier.normal]

claves.outliers.normales
```

```
## [1] 20
```

```r
data.frame.outliers.normales
```

```
##                 mpg cyl disp hp drat    wt qsec
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9
```

```r
nombres.outliers.normales
```

```
## [1] "Toyota Corolla"
```

```r
valores.outliers.normales
```

```
## [1] 33.9
```

```r
claves.outliers.extremos <- which(vector.es.outlier.extremo == TRUE)
data.frame.outliers.extremos <- as.data.frame(mydata.numeric[claves.outliers.extremos, ])
nombres.outliers.extremos <- row.names(mydata.numeric)[vector.es.outlier.extremo == TRUE]
valores.outliers.extremos <- columna[vector.es.outlier.extremo]

claves.outliers.extremos
```

```
## integer(0)
```

```r
data.frame.outliers.extremos
```

```
## [1] mpg  cyl  disp hp   drat wt   qsec
## <0 rows> (or 0-length row.names)
```

```r
nombres.outliers.extremos
```

```
## character(0)
```

```r
valores.outliers.extremos
```

```
## numeric(0)
```

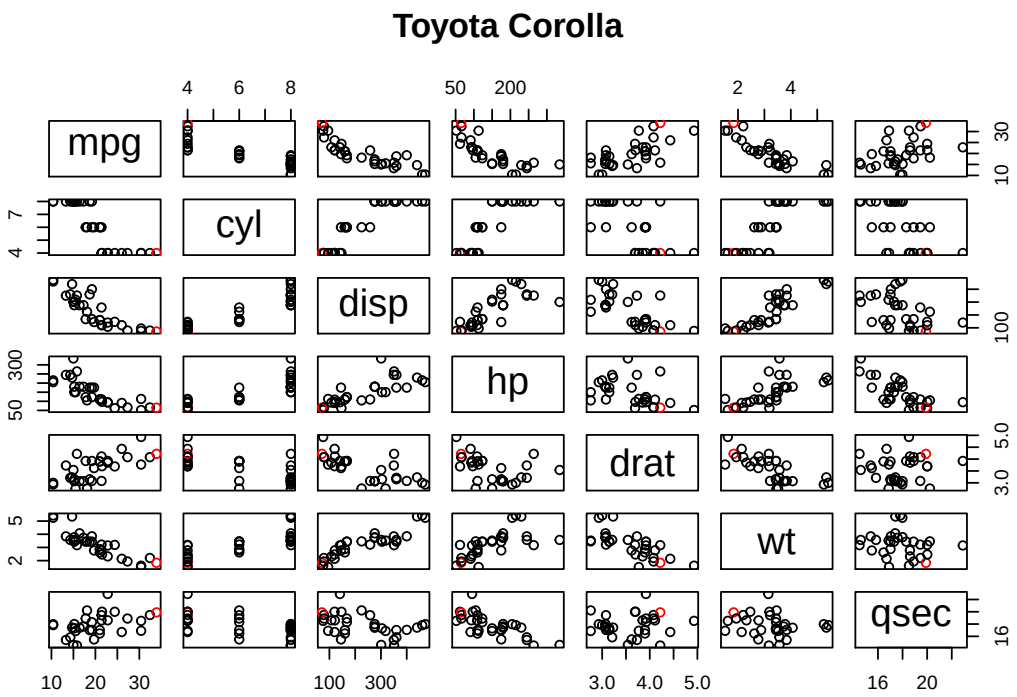
```r
##############################################################################
# Desviación de los outliers con respecto a la media de la columna
##############################################################################

# Construimos la variable:

# valores.normalizados.outliers.normales -> Contiene los valores normalizados de los
# outliers.
# Usad columna.scaled y (o bien vector.es.outlier.normal o bien claves.outliers.normales)

# Toyota Corolla
# 2.291272

# COMPLETAR

valores.normalizados.outliers.normales <- columna.scaled[vector.es.outlier.normal]
valores.normalizados.outliers.normales
```

```
## Toyota Corolla
##       2.291272
```

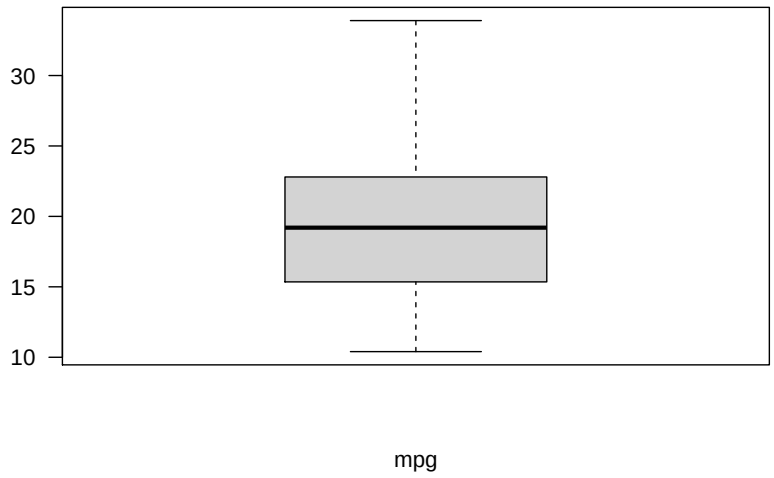
```
##############################################################################
# Plot
##############################################################################

# Mostramos en un plot los valores de los registros (los outliers se muestran en color
# rojo).
# Para ello, llamamos a la siguiente función:
# MiPlot_Univariate_Outliers (columna de datos, indices -claves numéricas- de outliers,
# nombre de columna).
# Lo hacemos con los outliers normales.

# COMPLETAR

MiPlot_Univariate_Outliers(mydata.numeric,
                           claves.outliers.normales,
                           nombres.outliers.normales)
```

```
##
## Número de datos: 32
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## FALSE FALSE FALSE FALSE FALSE FALSE
```

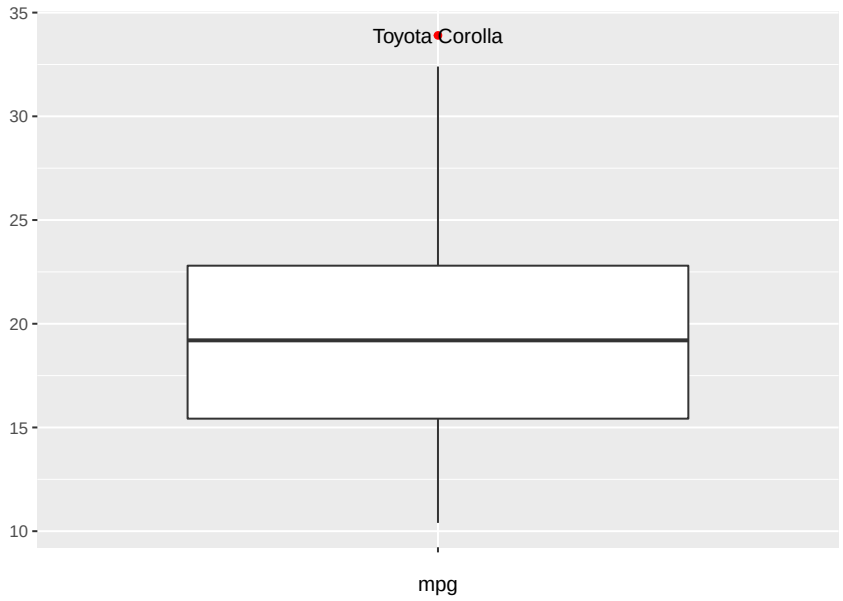
```
## MiPlot_Univariate_Outliers
```



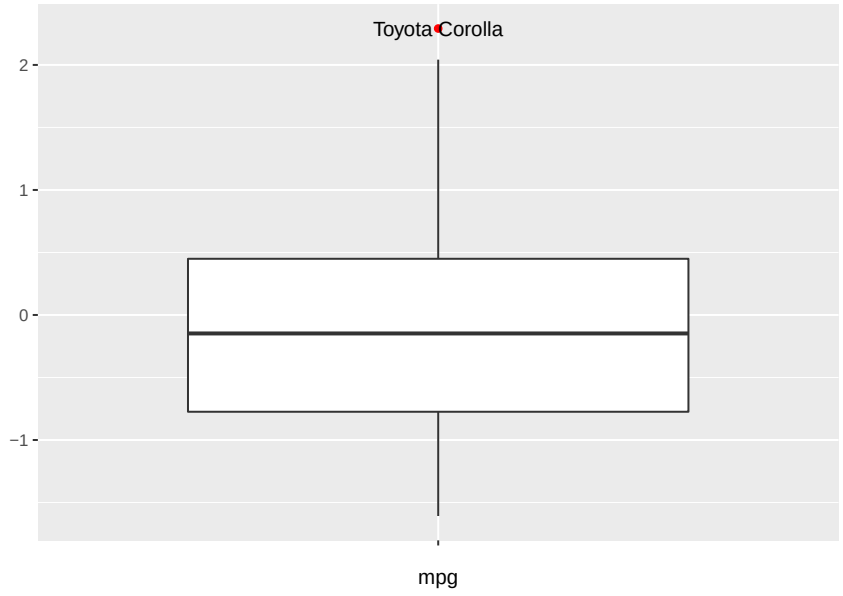
```
#############################################################################
# BoxPlot
#############################################################################

# Vemos el diagrama de caja.

# Para ello, llamaremos a la función boxplot, pero no muestra el outlier en la columna
# mpg, boxplot(columna, xlab=nombre.columna, main=nombre.mydata, las = 1)
# las = 1 all axis labels horizontal, range = 3 for exteme outliers.

# Para resolverlo, vemos el diagrama de caja con ggplot geom_boxplot
# Para ello, llamamos a la siguiente función
# MiBoxPlot_IQR_Univariate_Outliers = function (datos, indice.de.columna, coef = 1.5)

# Llamamos a la misma función pero con los datos normalizados
# Lo hacemos para resaltar que el Boxplot es el mismo ya que el poder de la normalización
# es que no afecta a la posición relativa de los datos

# COMPLETAR

boxplot(columna, xlab=nombre.columna, data=mydata.numeric, las = 1, range = 3)
```



```r
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric, indice.columna, coef = 1.5)
## MiBoxPlot_IQR_Univariate_Outliers
```




```r
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric.scaled, indice.columna, coef = 1.5)
## MiBoxPlot_IQR_Univariate_Outliers scaled
```




```r
##############################################################################
# Cómputo de los outliers IQR con funciones propias
##############################################################################

# En este apartado hacemos lo mismo que antes, pero llamando a funciones que están dentro
# de !Outliers_A3_Funciones_a_cargar_en_cada_sesion.R:
```

```
# vector_es_outlier_IQR -> devuelve un vector TRUE/FALSE
# vector.claves.outliers.IQR -> devuelve los índices de los outliers

vector.es.outlier.normal  = vector_es_outlier_IQR(mydata.numeric, indice.columna)
vector.es.outlier.extremo = vector_es_outlier_IQR(mydata.numeric, indice.columna, 3)

valores.outliers.normales = columna[vector.es.outlier.normal]
valores.outliers.extremos = columna[vector.es.outlier.extremo]

claves.outliers.normales = vector_claves_outliers_IQR (mydata.numeric, indice.columna)
claves.outliers.extremos = vector_claves_outliers_IQR (mydata.numeric, indice.columna, 3)

claves.outliers.normales
```

```
## [1] 20
```

```
valores.outliers.normales
```

```
## [1] 33.9
```

```
claves.outliers.extremos
```

```
## integer(0)
```

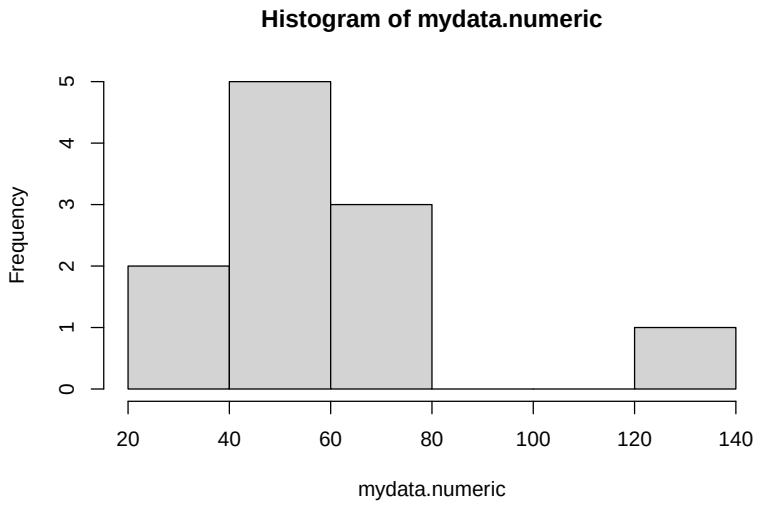
```
valores.outliers.extremos
```

```
## numeric(0)
```

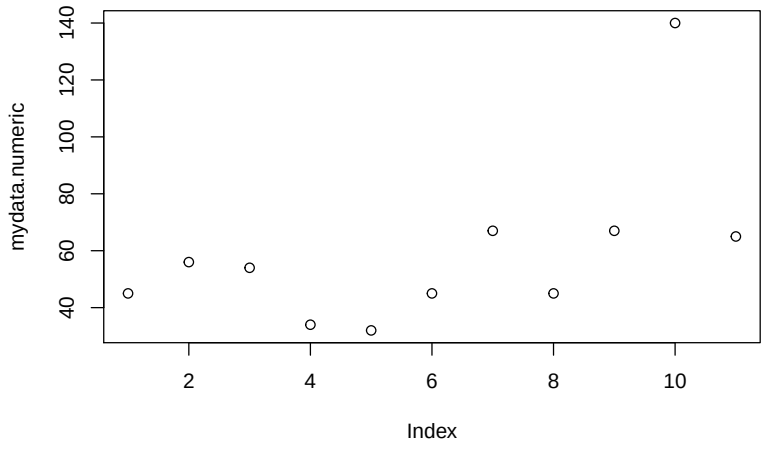
!MasterIA_GuionPracticas_Outliers_B2_1Variate_TestsEstadisticos

```
#############################################################################
# UNIVARIATE STATISTICAL OUTLIERS -> 1-variate Normal Distribution

# Grubbs' test. Normal 1-dim. 1 Único outlier.
# grubbs.test {outliers}

# Rosner's test. Normal 1-dim. <= k outliers.
# rosnerTest {EnvStats}

#############################################################################
# Conjuntos de datos
#############################################################################

datos.con.un.outlier = c(45,56,54,34,32,45,67,45,67,140,65)
datos.con.dos.outliers.masking = c(45,56,54,34,32,45,67,45,67,154,125,65)

mydata.numeric = datos.con.un.outlier
```

```
##############################################################################
# Test de Grubbs
##############################################################################

# Transparencia 85

##############################################################################
# datos.con.un.outlier
# Mostramos el histograma de mydata.numeric usando la función hist y un gráfico
# de puntos con la función plot.Observamos que hay un dato con un valor extremo.

# COMPLETAR

hist(mydata.numeric)
```




```
plot(mydata.numeric)
```



```r
# Aplicamos el test de Grubbs sobre datos.con.un.outlier.
# Usamos la función grubbs.test (two.sided = TRUE).
# Guardamos el resultado en test.de.Grubbs y vemos el p.value correspondiente.

# [1] 0.001126431

# Este resultado es significativo con los valores de alpha usuales 0.025, 0.01.

# COMPLETAR

test.de.Grubbs = grubbs.test(mydata.numeric, two.sided = TRUE)
test.de.Grubbs$p.value
```

```
## [1] 0.001126431
```

```r
# El test de Grubbs es significativo por lo que se concluye que hay un ÚNICO outlier.
# El valor que toma (140) los podríamos obtener a través de la función outlier del
# paquete outliers pero éste no nos dice cuál es el índice correspondiente (10).
# Por lo tanto, calculamos manualmente cuál es el índice de aquel registro
# que más se desvía de la media de la columna correspondiente.
# Tendremos que usar las funciones abs(valor absoluto), mean(media) y order (para ordenar).
# El resultado lo guardamos en las siguientes variables:

# indice.de.outlier.Grubbs
# valor.de.outlier.Grubbs

# [1] 10
# [1] 140

# COMPLETAR

indice.de.outlier.Grubbs = order(abs(mydata.numeric - mean(mydata.numeric)),
                                 decreasing = TRUE)[1]

valor.de.outlier.Grubbs = mydata.numeric[indice.de.outlier.Grubbs]

indice.de.outlier.Grubbs
```

```
## [1] 10
```

```r
valor.de.outlier.Grubbs
```

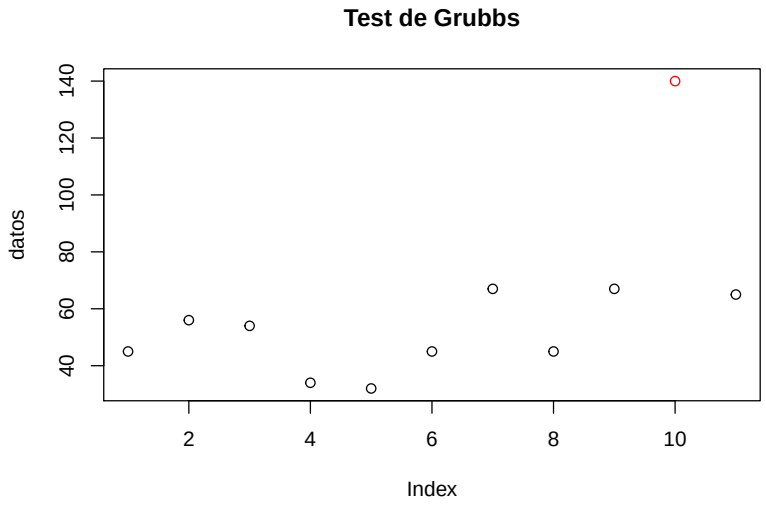
```
## [1] 140
```

```r
# Ahora que sabemos el índice del outlier, podemos usar la función
# MiPlot_Univariate_Outliers.
# Esta función muestra un plot similar al que ya hablámos mostrado, pero usa el color rojo
# para mostrar el outlier.
# Los parámetros son: el conjunto de datos, los índices de los outliers
# (sólo uno en este caso) y el título a mostrar.
# MiPlot_Univariate_Outliers = function (datos, indices_de_Outliers, titulo).
```

```r
# Resultado:

# Número de datos: 11
# ¿Quién es outlier?:
# FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE

# COMPLETAR

MiPlot_Univariate_Outliers(mydata.numeric, indice.de.outlier.Grubbs, "Test de Grubbs")
```

```
##
## Número de datos: 11
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

```r
## MiPlot_Univariate_Outliers
```

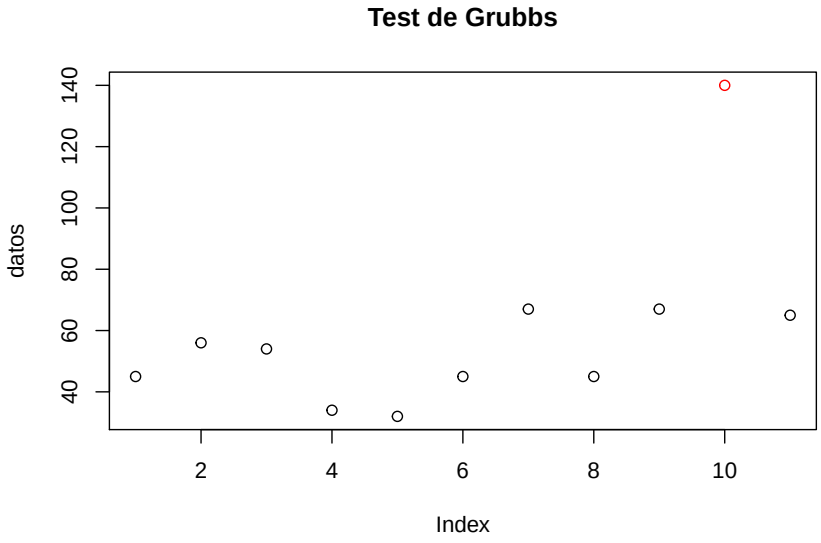



```r
################################################################################
# El mismo proceso anterior empaquetado en una función
################################################################################

# Llamamos a la función MiPlot_resultados_TestGrubbs
# MiPlot_resultados_TestGrubbs = function(datos).
# Esta función realiza todo el proceso de aplicar el test de Grubbs tal y como hemos hecho
# anteriormente. También muestra los resultados: para ello, la función llama directamente
# a MiPlot_Univariate_Outliers.
# El parámetro a pasar a la función MiPlot_resultados_TestGrubbs es el conjunto de datos.

# p.value: 0.001126431
# Índice de outlier: 10
# Valor del outlier: 140
# Número de datos: 11
# ¿Quién es outlier?:
# FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
```

```r
# COMPLETAR

MiPlot_resultados_TestGrubbs(mydata.numeric)
```

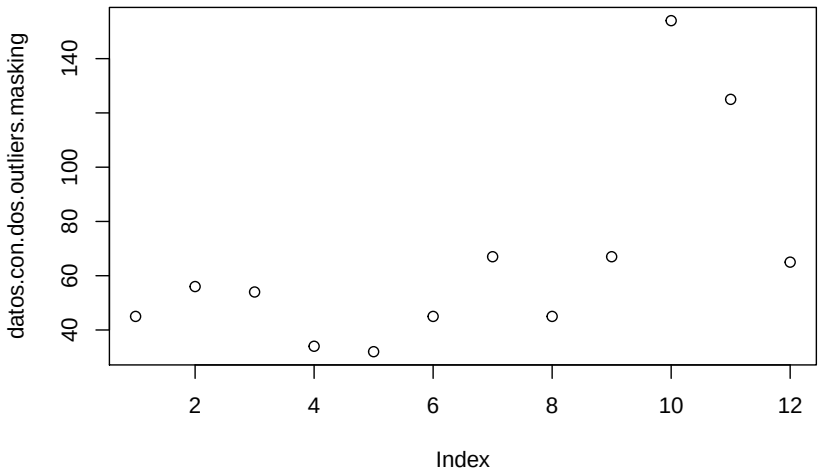
```
## p.value: 0.001126431
## Índice de outlier: 10
## Valor del outlier: 140
## Número de datos: 11
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

```r
## MiPlot_resultados_TestGrubbs
```




```r
##############################################################################
# Volvemos a aplicar el mismo proceso con los otros conjuntos de datos
##############################################################################

# Transparencia 88

##############################################################################
# datos.con.dos.outliers.masking
# Mostramos un gráfico de puntos con la función plot.
# Vemos que hay dos outliers.

# Aplicamos el test de Grubbs sobre datos.con.dos.outliers.masking.

# [1] 0.05614091

# El resultado no es significativo con ninguno de los valores de alpha usuales (<= 0.05).
# Sin embargo, hay dos outliers. (125, 154).
# La razón es que se ha producido un efecto de "masking".
# Ningún outlier es detectado por Grubbs.
```

```r
# COMPLETAR

plot(datos.con.dos.outliers.masking)
```




```r
test.de.Grubbs = grubbs.test(datos.con.dos.outliers.masking, two.sided = TRUE)
test.de.Grubbs$p.value
```

```
## [1] 0.05614091
```

```r
##############################################################################
# Test de Rosner
##############################################################################

# Hay tests para detectar un número exacto de k outliers, pero no son muy útiles.
# Mejor usamos un test para detectar un número menor o igual que k outliers (Rosner).

# Transparencia 90

# Aplicamos el Test de Rosner (rosnerTest) con k=4 sobre datos.con.dos.outliers.masking.
# Nos dará un aviso ocasionado por tener pocos datos.
# Guardamos el resultado en test.de.rosner .
# El test ordena los valores de mayor a menor distancia de la media y lanza el test
# de hipótesis para ver si hay menos de k=4 outliers.

# Imprimimos los siguientes campos:
# test.de.rosner$all.stats$Outlier
# Es un vector de 4 boolean.
# Nos indica si son considerados outliers los 4 valores que más se alejan de la media.
# En este caso:
# [1]  TRUE  TRUE FALSE FALSE
# Los dos primeros son TRUE y el resto FALSE => El test indica que hay dos outliers
```

```r
# test.de.rosner$all.stats$Obs.Num
# Es un vector con los cuatro índices de los 4 valores que más se alejan de la media.
# En este caso:
# [1]  10    11   5     4

# Construimos el vector con los índices de los que son outliers (10, 11)
# y se lo pasamos como parámetro a la función.
# MiPlot_Univariate_Outliers
# MiPlot_Univariate_Outliers = function (datos, indices_de_Outliers, titulo)

# Número de datos: 12
# ¿Quién es outlier?:
# FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE

# COMPLETAR

test.de.rosner = rosnerTest(datos.con.dos.outliers.masking, k = 4)
test.de.rosner$all.stats$Outlier
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

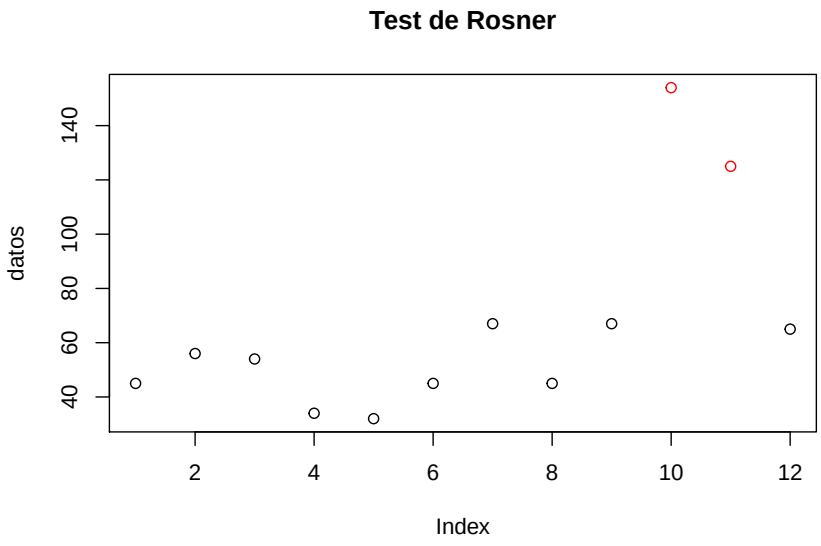
```r
test.de.rosner$all.stats$Obs.Num
```

```
## [1] 10 11  5  4
```

```r
MiPlot_Univariate_Outliers(datos.con.dos.outliers.masking, c(10, 11), "Test de Rosner")
```

```
##
## Número de datos: 12
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE
```

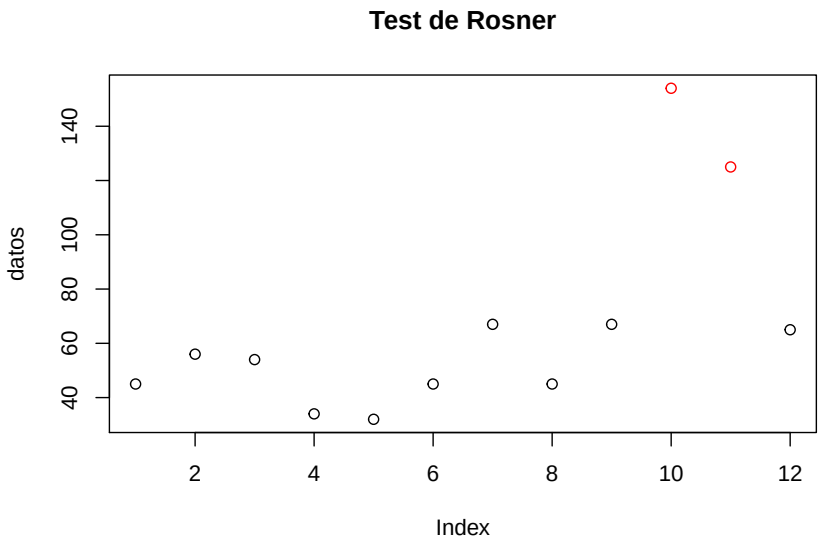
```r
## MiPlot_Univariate_Outliers
```



```
#############################################################################
# La función MiPlot_resultados_TestRosner = function(datos)
# hace directamente las anteriores tareas, es decir, lanza el test y dibuja el plot.
# Lanzamos esta función con el dataset datos.con.dos.outliers.masking
# y comprobamos que ofrece los resultados vistos anteriormente.

# COMPLETAR

MiPlot_resultados_TestRosner(datos.con.dos.outliers.masking)
```

```
##
## Test de Rosner
## Índices de las k-mayores desviaciones de la media: 10 11 5 4
## De las k mayores desviaciones, ¿Quién es outlier? TRUE TRUE FALSE FALSE
## Los índices de los outliers son: 10 11
## Los valores de los outliers son: 154 125
## Número de datos: 12
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE
```

```
## MiPlot_resultados_TestRosner
```

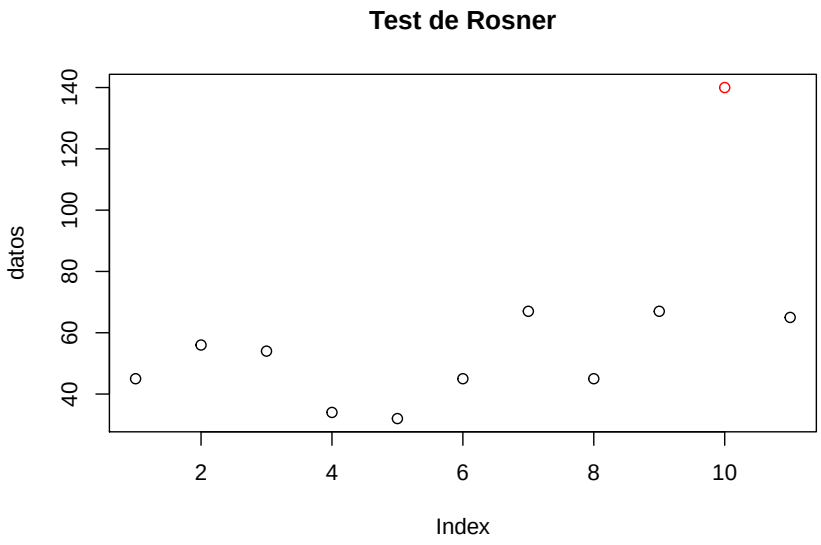



```
#############################################################################
# Para ver el comportamiento del Test de Rosner con el conjunto de datos inicial
# lanzamos la función MiPlot_resultados_TestRosner con k=4 sobre datos.con.un.outlier.

# Test de Rosner
# Índices de las k-mayores desviaciones de la media: 10 5 4 7
# De las k mayores desviaciones, ¿Quién es outlier? TRUE FALSE FALSE FALSE
# Los índices de los outliers son: 10
# Los valores de los outliers son: 140
# Número de datos: 11
```

```
# ¿Quién es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE

# El test indica que sólo hay un outlier.

# COMPLETAR

MiPlot_resultados_TestRosner(mydata.numeric)
```

```
##
## Test de Rosner
## Índices de las k-mayores desviaciones de la media: 10 5 4 7
## De las k mayores desviaciones, ¿Quién es outlier? TRUE FALSE FALSE FALSE
## Los índices de los outliers son: 10
## Los valores de los outliers son: 140
## Número de datos: 11
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

```
## MiPlot_resultados_TestRosner
```

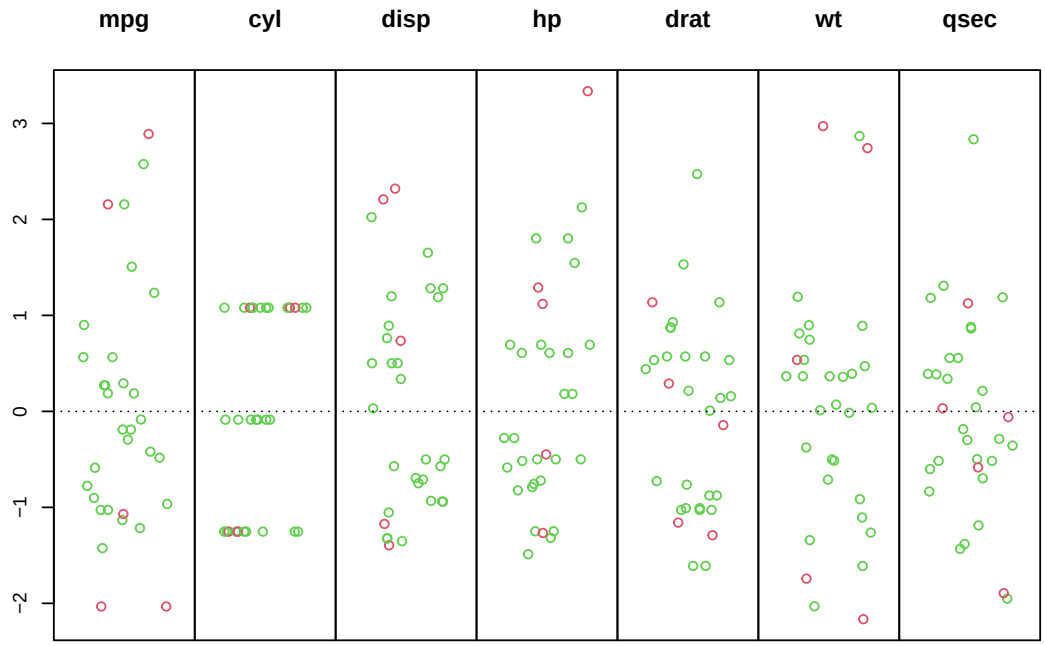


!MasterIA_GuionPracticas_Outliers_C1_MultiVariate_Mahalanobis

```
##############################################################################
# MULTIVARIATE STATISTICAL OUTLIERS -> Multivariate Normal Distribution --> Mahalanobis
##############################################################################

# Los outliers son respecto a un conjunto de variables.
# Un registro será un outlier porque tenga un valor anómalo en alguna variable
# o porque tenga una combinación anómala de valores.
```

```r
# Necesita:
# mydata.numeric
# mydata.numeric.scaled

# Trabajamos sobre mtcars[, -c(8:11)]

mydata.numeric = mtcars[, -c(8:11)]
mydata.numeric.scaled = scale(mydata.numeric)

################################################################################
# Paquete mvoutlier
################################################################################

# Obtención de los outliers multivariantes

# Transparencia 95

# Calcula los outliers calculando las distancias de Mahalanobis y usando la
# aproximación de la Chi cuadrado.
# La estimación de la matriz de covarianzas es la estimación robusta según MCD.
# No hay que normalizar los datos ya que la distancia de Mahalanobis está
# diseñada, precisamente para evitar el problema de la escala.

# uni.plot genera el gráfico similar a MiPlot_Univariate_Outliers con todas las columnas.
# Además, devuelve en $outliers los índices de los outliers.

# Establecemos los valores de significación.
# alpha.value.penalizado es para tener en cuenta el error FWER.

alpha.value = 0.05
alpha.value.penalizado = 1 - (1 - alpha.value) ^ (1 / nrow(mydata.numeric))

# Transparencia 91

# Establecemos la semilla para el método iterativo que calcula MCD
# IMPORTANTE: Para que el resultado sea el mismo en todas las ejecuciones, siempre
# hay que establecer la semilla antes de lanzar la función correspondiente.

set.seed(12)

# Llamamos a uni.plot del paquete mvoutlier con symb=FALSE, alpha = alpha.value.penalizado.
# Guardamos el resultado en la variable mvoutlier.plot.
# Esta función calcula los outliers MULTIVARIANTES según la distancia de Mahalanobis
# considerando la estimación robusta de la matriz de covarianzas -MCD- y la estimación
# robusta de la media de cada variable.
# También imprime un plot 1-dimensional para ver los valores que toman los outliers en
# cada atributo pero el plot no imprime las etiquetas de los outliers.

# Nota: Es posible que haya que instalar el paquete pcaPP para que se pueda ejecutar
# uni.plot.

X11()
```

```r
# COMPLETAR

mvoutlier.plot = uni.plot(mydata.numeric, symb=FALSE, alpha = alpha.value.penalizado)
```




```r
##############################################################################
# Análisis de los outliers

# Vamos a ver las variables que más influyen en la designación de los outliers.
# a) Viendo el valor normalizado sobre cada variable para ver cuánto se desvía de la media.
# Pero esto no es suficiente ya que no es fácil apreciar interacciones entre atributos.
# b) Gráficamente, con un biplot sobre las componentes principales.
# El Biplot permite ver las dimensiones importantes que influyen en la designación de
# los outliers.

# Construimos las variables.
# is.MCD.outlier
# numero.de.outliers.MCD
# que será un vector TRUE/FALSE que nos dice si cada dato es o no un outlier.
# Para ello, accedemos a mvoutlier.plot$outliers.
# Contamos el número total de outliers y lo guardamos en la variable
# numero.de.outliers.MCD.
# Debe salir lo siguiente:

# is.MCD.outlier
# Mazda RX4       Mazda RX4 Wag          Datsun 710   ......
# FALSE               FALSE               FALSE       ......
# ......
```

```
# numero.de.outliers.MCD
# [1] 10

# COMPLETAR

is.MCD.outlier = mvoutlier.plot$outliers
numero.de.outliers.MCD = sum(is.MCD.outlier)
is.MCD.outlier
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##               FALSE               FALSE               FALSE               FALSE
##   Hornet Sportabout             Valiant          Duster 360           Merc 240D
##               FALSE               FALSE               FALSE               FALSE
##            Merc 230            Merc 280           Merc 280C          Merc 450SE
##                TRUE               FALSE               FALSE               FALSE
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##               FALSE               FALSE                TRUE                TRUE
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##                TRUE                TRUE                TRUE                TRUE
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##               FALSE               FALSE               FALSE               FALSE
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##               FALSE               FALSE               FALSE                TRUE
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##               FALSE                TRUE                TRUE               FALSE
```

```
numero.de.outliers.MCD
```

```
## [1] 10
```

```
# Calculamos los índices de los outliers multivariantes en la variable
# indices.de.outliers.multivariantes.MCD
# y los mostramos en pantalla. Debe salir lo siguiente:

# indices.de.outliers.multivariantes.MCD
# Merc 230  Cadillac Fleetwood Lincoln Continental   Chrysler Imperial          Fiat 128
# 9                  15                  16                  17                  18
# Honda Civic      Toyota Corolla        Lotus Europa        Ferrari Dino   Maserati Bora
# 19

# Vemos qué outliers son multivariantes "puros", es decir, que NO son 1 variantes
# con respecto a ninguna columna. Estos outliers multivariantes son interesantes
# ya que nos indican que no son outliers porque una de sus columnas tenga un valor
# extremo, sino porque hay alguna combinación anómala de valores de columnas.

# Por tanto, en primer lugar, debemos obtener los índices (claves) de aquellos registros
# que son outliers IQR en alguna de las columnas.
# Para ello, basta usar la función vector_claves_outliers_IQR_en_alguna_columna
# y construimos la variable indices.de.outliers.en.alguna.columna

# Con las variables indices.de.outliers.en.alguna.columna y
# indices.de.outliers.multivariantes.MCD construimos las siguientes variables:
# indices.de.outliers.multivariantes.MCD.pero.no.1variantes (debe usar setdiff).
```

```
# nombres.de.outliers.multivariantes.MCD.pero.no.1variantes (debe usar rownames).

# Debe salir lo siguiente:

# indices.de.outliers.multivariantes.MCD
# Merc 230  Cadillac Fleetwood Lincoln Continental   Chrysler Imperial          Fiat 128
# 9                  15                  16                  17                  18
# Honda Civic      Toyota Corolla        Lotus Europa        Ferrari Dino   Maserati Bora
# 19                  20                  28                  30                  31

# indices.de.outliers.multivariantes.MCD.pero.no.1variantes
# [1] 18 19 28 30

# nombres.de.outliers.multivariantes.MCD.pero.no.1variantes
# [1] "Fiat 128"     "Honda Civic"  "Lotus Europa" "Ferrari Dino"

# COMPLETAR

indices.de.outliers.multivariantes.MCD <- which(is.MCD.outlier==TRUE)
indices.de.outliers.en.alguna.columna <- vector_claves_outliers_IQR_en_alguna_columna(
  mydata.numeric, coef = 1.5)

indices.de.outliers.multivariantes.MCD.pero.no.1variantes <- setdiff(
  indices.de.outliers.multivariantes.MCD, indices.de.outliers.en.alguna.columna)

data.frame.de.outliers.multivariantes.MCD.pero.no.1variantes <- as.data.frame(
  mydata.numeric[indices.de.outliers.multivariantes.MCD.pero.no.1variantes,])

nombres.de.outliers.multivariantes.MCD.pero.no.1variantes <- row.names(
  data.frame.de.outliers.multivariantes.MCD.pero.no.1variantes)

indices.de.outliers.multivariantes.MCD
```

```
##            Merc 230  Cadillac Fleetwood Lincoln Continental   Chrysler Imperial
##                   9                  15                  16                  17
##            Fiat 128         Honda Civic      Toyota Corolla        Lotus Europa
##                  18                  19                  20                  28
##        Ferrari Dino       Maserati Bora
##                  30                  31
```

```
indices.de.outliers.multivariantes.MCD.pero.no.1variantes
```

```
## [1] 18 19 28 30
```

```
nombres.de.outliers.multivariantes.MCD.pero.no.1variantes
```

```
## [1] "Fiat 128"     "Honda Civic"  "Lotus Europa" "Ferrari Dino"
```

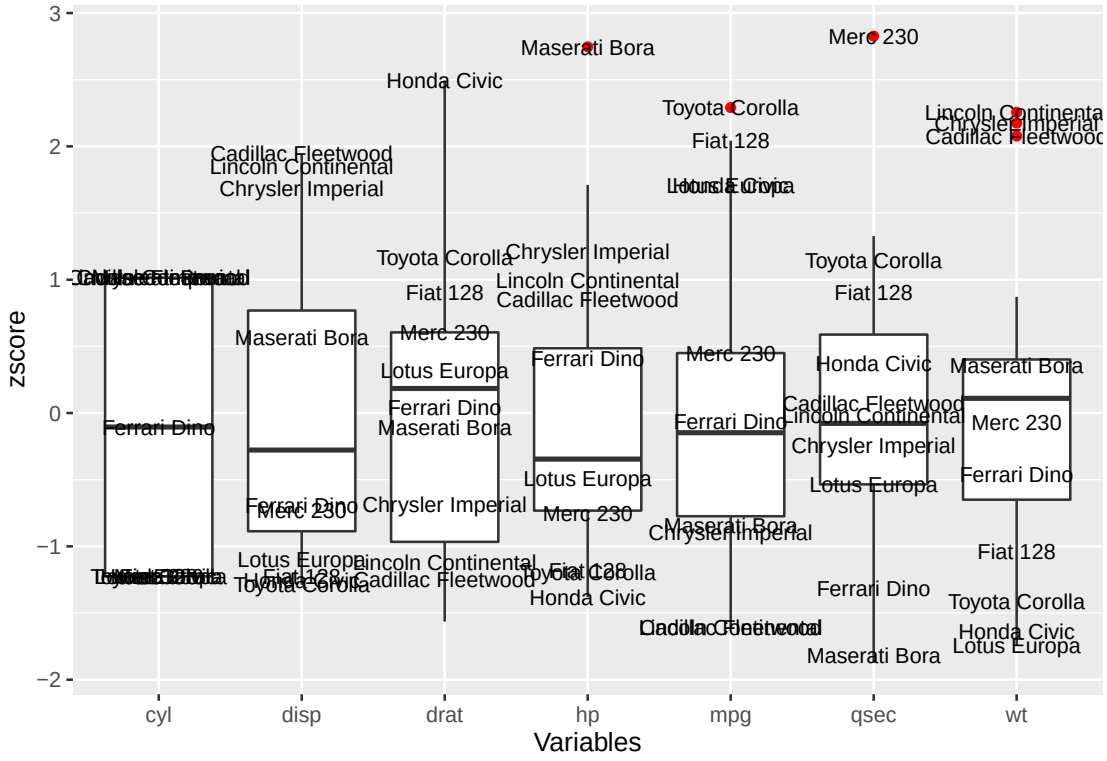
```
# ¿Cuál es el valor normalizado de cada outlier, es decir,
# ¿Cuánto se desvía de la media de cada columna?
# Esta desviación ya se ha mostrado antes al llamar a uni.plot,
```

```
# pero sólo se muestran los outliers como puntos rojos.
# Al no tener las etiquetas, no sabemos cuáles son los valores de los outliers
# en cada columna.

# Construimos una tabla numérica data.frame.solo.outliers que muestre los valores
# normalizados de los outliers en todas las columnas.
# Para ello, usamos mydata.numeric.scaled y is.MCD.outlier:

# mpg        cyl       disp         hp        drat          wt        qsec
# Merc 230             0.44954345 -1.2248578 -0.7255351 -0.7538702  0.60491932 -0.06873063
# Cadillac Fleetwood  -1.60788262  1.0148821  1.9467538  0.8504968 -1.24665983  2.07750476
# Lincoln Continental -1.60788262  1.0148821  1.8499318  0.9963483 -1.11574009  2.25533570
# Chrysler Imperial   -0.89442035  1.0148821  1.6885616  1.2151256 -0.68557523  2.17459637
# Fiat 128             2.04238943 -1.2248578 -1.2265893 -1.1768396  0.90416444 -1.03964665
# Honda Civic          1.71054652 -1.2248578 -1.2507948 -1.3810318  2.49390411 -1.63752651
# Toyota Corolla       2.29127162 -1.2248578 -1.2879099 -1.1914248  1.16600392 -1.41268280
# Lotus Europa         1.71054652 -1.2248578 -1.0942658 -0.4913374  0.32437703 -1.74177223
# Ferrari Dino        -0.06481307 -0.1049878 -0.6916474  0.4129422  0.04383473 -0.45709704
# Maserati Bora       -0.84464392  1.0148821  0.5670394  2.7465668 -0.10578782  0.36051645

# COMPLETAR

data.frame.solo.outliers <- as.data.frame(
  mydata.numeric.scaled[indices.de.outliers.multivariantes.MCD,])

data.frame.solo.outliers
```

```
##                           mpg        cyl       disp         hp        drat
## Merc 230             0.44954345 -1.2248578 -0.7255351 -0.7538702  0.60491932
## Cadillac Fleetwood  -1.60788262  1.0148821  1.9467538  0.8504968 -1.24665983
## Lincoln Continental -1.60788262  1.0148821  1.8499318  0.9963483 -1.11574009
## Chrysler Imperial   -0.89442035  1.0148821  1.6885616  1.2151256 -0.68557523
## Fiat 128             2.04238943 -1.2248578 -1.2265893 -1.1768396  0.90416444
## Honda Civic          1.71054652 -1.2248578 -1.2507948 -1.3810318  2.49390411
## Toyota Corolla       2.29127162 -1.2248578 -1.2879099 -1.1914248  1.16600392
## Lotus Europa         1.71054652 -1.2248578 -1.0942658 -0.4913374  0.32437703
## Ferrari Dino        -0.06481307 -0.1049878 -0.6916474  0.4129422  0.04383473
## Maserati Bora       -0.84464392  1.0148821  0.5670394  2.7465668 -0.10578782
##                           wt        qsec
## Merc 230            -0.06873063  2.82675459
## Cadillac Fleetwood   2.07750476  0.07344945
## Lincoln Continental  2.25533570 -0.01608893
## Chrysler Imperial    2.17459637 -0.23993487
## Fiat 128            -1.03964665  0.90727560
## Honda Civic         -1.63752651  0.37564148
## Toyota Corolla      -1.41268280  1.14790999
## Lotus Europa        -1.74177223 -0.53093460
## Ferrari Dino        -0.45709704 -1.31439542
## Maserati Bora        0.36051645 -1.81804880
```

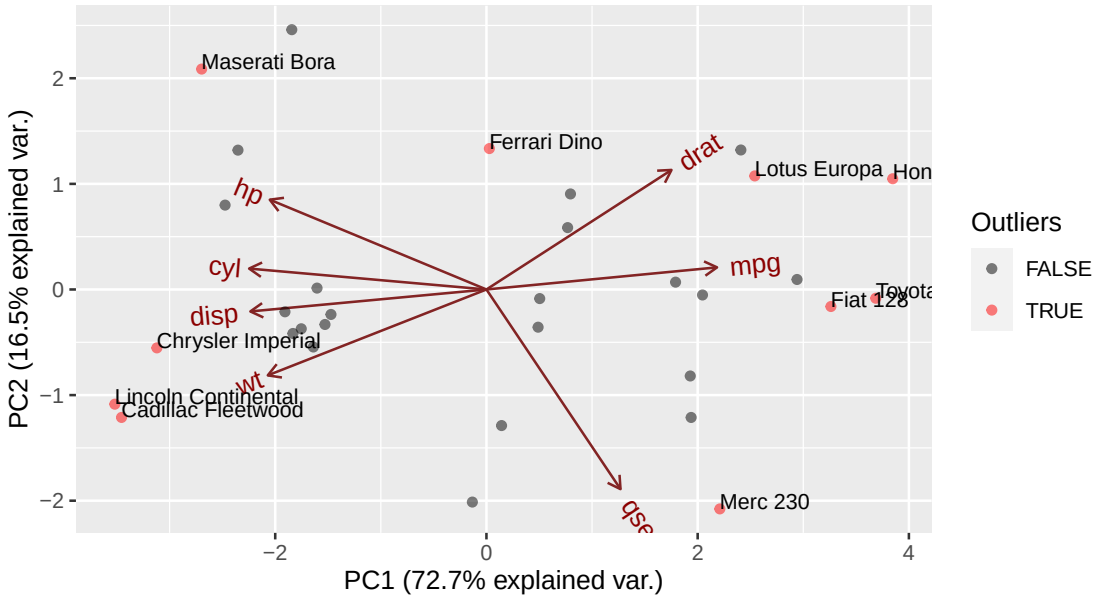
```
# Mostramos los boxplots de forma conjunta con las etiquetas de los outliers.
# Para ello llamamos a la función MiBoxPlot_juntos pasando como parámetro is.MCD.outlier.
# MiBoxPlot_juntos = function (datos, vector_TF_datos_a_incluir)
```

```
# COMPLETAR

MiBoxPlot_juntos(mydata.numeric, is.MCD.outlier)
```




```
# Transparencia 79  (Biplot)

# El BoxPlot conjunto nos informa sobre los valores extremos que hay en cada variable.
# Puede apreciarse que casi todos los outliers multivariate corresponden a outliers
# univariate.
# Las únicas excepciones son Fiat 128 y Ferrari Dino, aunque Fiat 128 es casi un outlier
# en mpg.

# El BiPlot nos muestra también esta información, junto con las correlaciones
# entre variables.
# Los puntos mostrados son resultados de proyecciones de n dimensiones a 2, por lo que
# sólo es una representación aproximada (mejor cuanto mayor sea la suma de los porcentajes
# que aparecen como componentes principales PC1 y PC2).
# Llamamos a la función MiBiPlot_Multivariate_Outliers
# MiBiPlot_Multivariate_Outliers = function (datos, vectorTFoutliers, titulo)

# COMPLETAR

MiBiPlot_Multivariate_Outliers(mydata.numeric, is.MCD.outlier, "")
```

```
## MiBiPlot_Multivariate_Outliers
```




```
# El BiPlot muestra claramente que Ferrari Dino no es outlier univariate en ninguna
# variable.
# (no está en el extremo delimitado por los vectores correspondientes a las variables)
# Posiblemente sea un outlier multivariate debido a la combinación anormal de varias
# variables.

# Vamos a construir una matriz con los gráficos de dispersión obtenidos al cruzar todas
# las variables.
# Y vamos a destacar en rojo el dato correspondiente a Ferrari Dino.
# Para ello, obtenemos el índice de Ferrari Dino usando las funciones which y rownames
# y llamamos a la función MiPlot_Univariate_Outliers
# MiPlot_Univariate_Outliers = function (datos, indices_de_Outliers, titulo)
# El parámetro indices_de_Outliers únicamente contendrá el índice del Ferrari Dino.
# Puede apreciarse que no hay una combinación clara de 2 variables que hagan del Ferrari
# un outlier.
# Es posible que intervengan más de dos variables.
# Efectivamente, si observamos la tabla data.frame.solo.outliers
# parece ser que consigue una aceleración qsec muy buena -1.3
# (bastante cercana a la mayor -> Maserati Bora -1.8)
# con una potencia hp normal 0.4 (Maserati 2.7). Tener un peso wt ligero -0.4 seguramente
# es un factor decisivo (Maserati 0.3).
# La combinación peso, aceleración, hp es lo que hace de Ferrari Dino un outlier
# multivariate.

# COMPLETAR
```

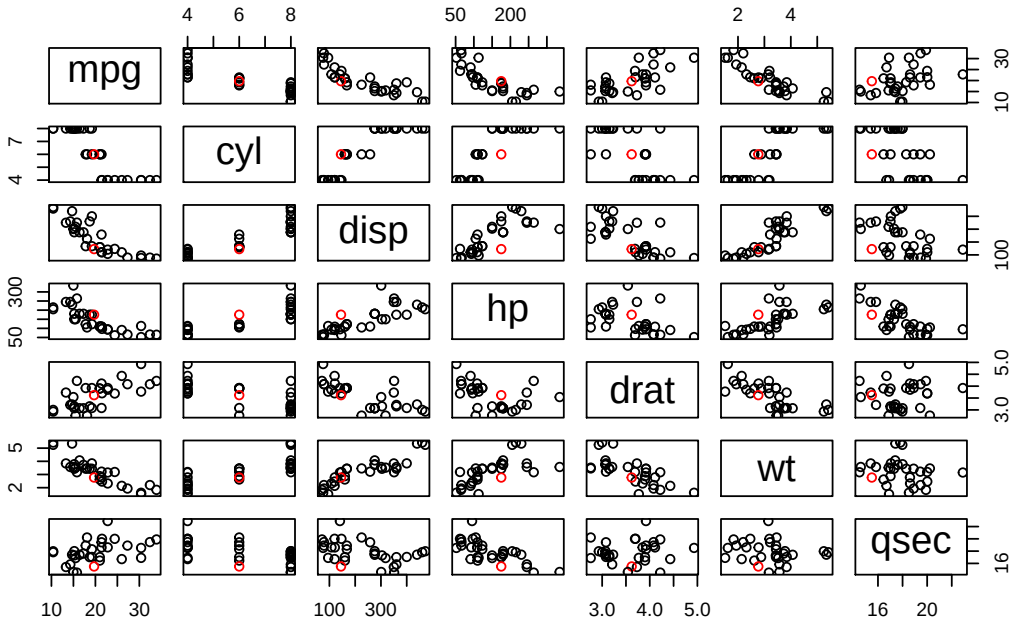
```
indices.de.outliers <- as.data.frame(indices.de.outliers.multivariantes.MCD)
indices.de.outliers.en.Ferrari.Dino <- indices.de.outliers["Ferrari Dino",]
indices.de.outliers.en.Ferrari.Dino
```

```
## [1] 30
```

```
MiPlot_Univariate_Outliers(mydata.numeric, indices.de.outliers.en.Ferrari.Dino, "")
```

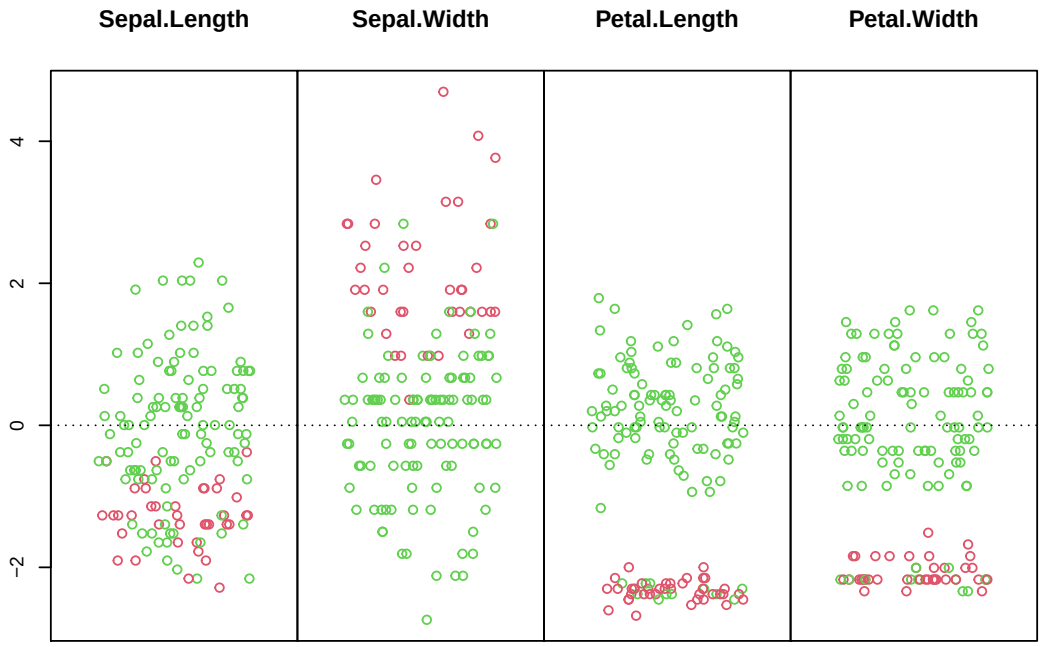
```
##
## Número de datos: 32
## ¿Quién es outlier?:
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## FALSE FALSE FALSE TRUE FALSE FALSE
```

```
## MiPlot_Univariate_Outliers
```



!MasterIA_GuionPracticas_Outliers_D1_LOF

```
##############################################################################
# MULTIVARIATE STATISTICAL OUTLIERS -> LOF
##############################################################################

# Los outliers son respecto a un conjunto de variables.

#############################################################################
# Lectura de valores y Preprocesamiento
#############################################################################

# Tanto LOF como clustering usan distancias entre registros, por lo que habrá
# que trabajar sobre los datos previamente normalizados.

# Construimos las siguientes variables:

# mis.datos.numericos -> Contendrá las columnas numéricas de iris, es decir, iris [1:4].
# mis.datos.numericos.normalizados -> Contendrá los datos normalizados.
# Asignamos como nombres de filas de mis.datos.numericos.normalizados los mismos nombres
# de filas que mis.datos.numericos.

# Ampliación: Utilice la función is.numeric y sapply para construir automáticamente un
# data frame con las columnas numéricas de otro data frame.

mis.datos.originales = iris
mis.datos.numericos  = mis.datos.originales[,1:4]
mis.datos.numericos  = mis.datos.originales[,sapply(mis.datos.originales, is.numeric)]
mis.datos.numericos.normalizados = scale(mis.datos.numericos)
rownames(mis.datos.numericos.normalizados) = rownames(mis.datos.numericos)

#############################################################################

# Transparencia 106

# Para comprobar que el método de Mahalanobis no es aplicable,
# obtenga las variables is.MCD.outlier y numero.de.outliers.MCD
# tal y como se hizo en el script anterior (hay que tener cargada la librería mvoutlier)
# Observe que hay un número muy elevado de outliers (50) y además con valores
# de Petal.Length y Petal.Width muy similares.
# Realmente no son outliers sino que forman un grupo homogéneo.

# COMPLETAR

alpha.value = 0.05
alpha.value.penalizado = 1 - (1 - alpha.value) ^ (1 / nrow(mis.datos.numericos))

set.seed(12)
X11()

mvoutlier.plot = uni.plot(mis.datos.numericos, symb=FALSE, alpha = alpha.value.penalizado)
```



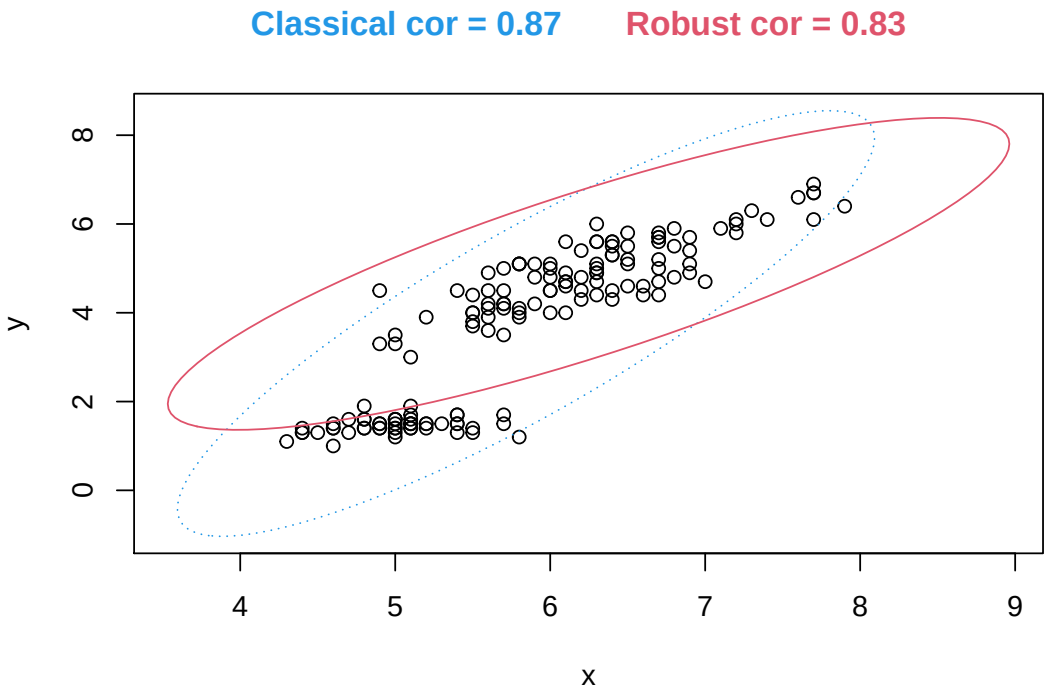

```
is.MCD.outlier = mvoutlier.plot$outliers
numero.de.outliers.MCD = sum(is.MCD.outlier)
is.MCD.outlier
```

```
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [49]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
numero.de.outliers.MCD
```

```
## [1] 50
```

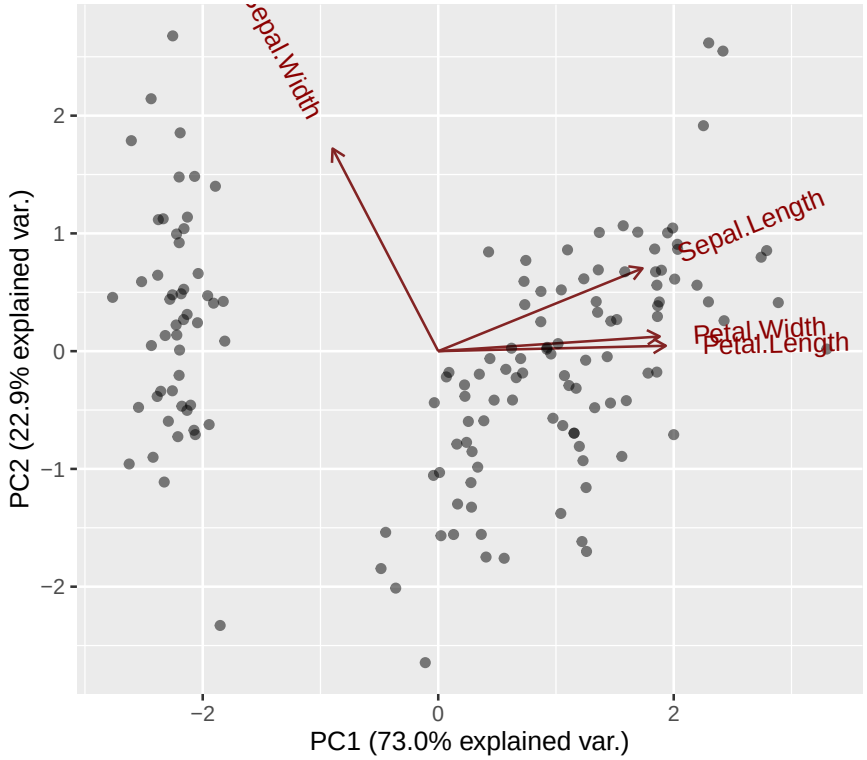
```
# Ejecute también lo siguiente:

X11()
corr.plot(mis.datos.numericos[,1], mis.datos.numericos[,3])
```




```
## $cor.cla
## [1] 0.8717538
##
## $cor.rob
## [1] 0.8339014
```

```
# El gráfico nos muestra un gráfico de dispersión al cruzar las variables 1 y 3.
# Vemos que hay dos grupos bien definidos de datos.
# Los puntos que hay entre ellos deberían ser marcados como outliers.
# Usando la distancia de Mahalanobis clásica (azul) el elipsoide
# contiene a ambos grupos por lo que los puntos que hubiese entre ellos no serían outliers.
# Usando la distancia de Mahalanobis construida con la estimación robusta de la matriz
# de covarianzas y las correspondientes medias,
# el elipsoide (rojo) se construye con el grupo de datos
# más numeroso y todos los datos del otro grupo se marcan como outliers.

# También podemos mostrar un BiPlot llamando a la función MiBiplot sobre
# mis.datos.numericos.
# El gráfico mostrado es una simplificación ya que ahora estamos mostrando las cuatro
# variables conjuntamente en un gráfico 2 dimensional (Transparencia 72).
# Podemos apreciar que hay dos nubes de puntos bien separadas.

# Así pues, el método de detección de outliers usando la distancia de Mahalanobis
# no es adecuado.



MiBiplot(mis.datos.numericos)
```

```
## MiBiplot
```

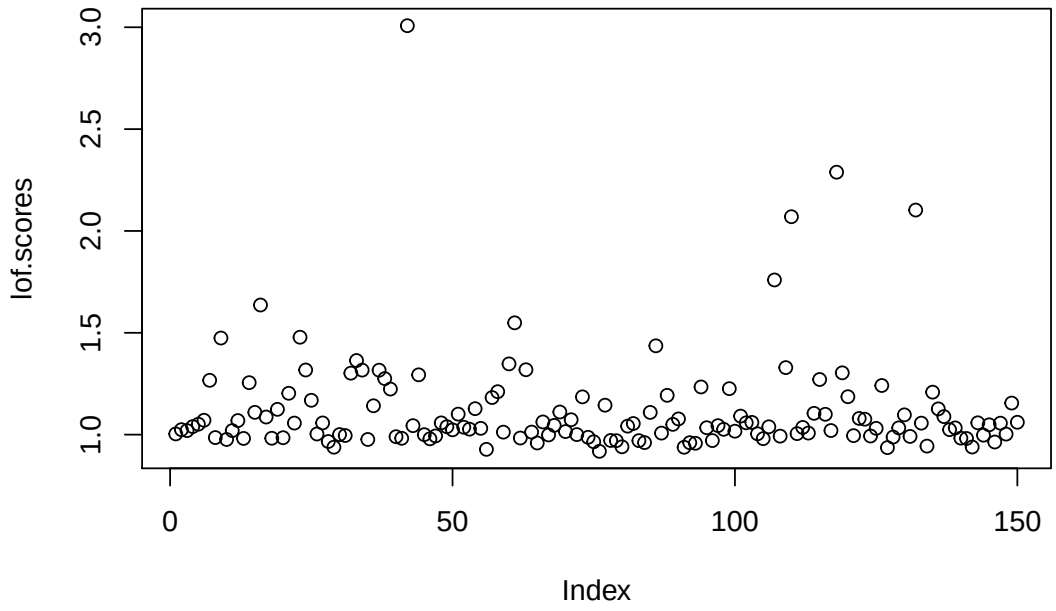



```
##############################################################################
# DISTANCE BASED OUTLIERS (LOF)
##############################################################################

# Transparencia 124

numero.de.vecinos.lof = 5

# Establecemos el número de vecinos a considerar numero.de.vecinos.lof = 5 y llamamos a
# la función lofactor pasándole como primer parámetro el conjunto de datos normalizados
# y como parámetro k el valor de numero.de.vecinos.lof.
# Esta función devuelve un vector con los scores de LOF de todos los registros.
# Lo llamamos lof.scores
# [1] 1.0036218 1.0244637 1.0198058 1.0394019 ......

# Hacemos un plot de los resultados (basta llamar a la función plot sobre lof.scores)
# para ver los scores obtenidos por LOF.
# Podemos apreciar que hay 4 valores de lof notablemente míos altos que el resto.
# Así pues, establecemos la variable siguiente:
# numero.de.outliers = 4

# Ordenamos los lof.scores y obtenemos los índices de los registros ordenados según
# el lof.score indices.de.lof.outliers.ordenados.
# [1]  42 118 132 110 107  16  61  23  ......

# Seleccionamos los 4 primeros y los almacenamos en indices.de.lof.top.outliers.
```

```
# [1]  42 118 132 110

# Construimos un vector is.lof.outlier de TRUE/FALSE que nos dice si cada registro
# de los datos originales es o no un outlier. Para ello, debemos usar la función
# rownames sobre el dataset y el operador %in% sobre indices.de.lof.top.outliers.
# is.lof.outlier
# [1] FALSE FALSE FALSE FALSE FALSE ......

# Mostramos un Biplot de los outliers llamando a la función MiBiPlot_Multivariate_Outliers
# MiBiPlot_Multivariate_Outliers = function (datos, vectorTFoutliers, titulo)

# Tal vez, el dato más interesante sea el 42 ya que no parece que sea un outlier
# univariante (luego lo comprobaremos).

# COMPLETAR

lof.scores = lofactor(mis.datos.numericos.normalizados, numero.de.vecinos.lof)
lof.scores
```

```
##   [1] 1.0036218 1.0244637 1.0198058 1.0394019 1.0513006 1.0705307 1.2667824
##   [8] 0.9856413 1.4742692 0.9763678 1.0198334 1.0690899 0.9814612 1.2552971
##  [15] 1.1095385 1.6364042 1.0865227 0.9822760 1.1242898 0.9849325 1.2030882
##  [22] 1.0567536 1.4782740 1.3173590 1.1686321 1.0031977 1.0574051 0.9663117
##  [29] 0.9383734 1.0004297 0.9958319 1.3018241 1.3638789 1.3168633 0.9763678
##  [36] 1.1416193 1.3164521 1.2752170 1.2237788 0.9899075 0.9821934 3.0076370
##  [43] 1.0440598 1.2937075 0.9998983 0.9794853 0.9935653 1.0576509 1.0389251
##  [50] 1.0236642 1.1008010 1.0363400 1.0268909 1.1274917 1.0309196 0.9278019
##  [57] 1.1822118 1.2109074 1.0122421 1.3474459 1.5490514 0.9834815 1.3186566
##  [64] 1.0125729 0.9590798 1.0623705 0.9986236 1.0454868 1.1111003 1.0151137
##  [71] 1.0732311 1.0002806 1.1854471 0.9869137 0.9651281 0.9181960 1.1445896
##  [78] 0.9714175 0.9704947 0.9403407 1.0417554 1.0545556 0.9705678 0.9610218
##  [85] 1.1092103 1.4361000 1.0075801 1.1930267 1.0504362 1.0772411 0.9380292
##  [92] 0.9606317 0.9576519 1.2342022 1.0339427 0.9709630 1.0442302 1.0262605
##  [99] 1.2257850 1.0168971 1.0909658 1.0580851 1.0601194 1.0040903 0.9802590
## [106] 1.0385085 1.7593911 0.9926957 1.3291887 2.0699369 1.0050052 1.0354129
## [113] 1.0077273 1.1046630 1.2707539 1.0999640 1.0202272 2.2884244 1.3034166
## [120] 1.1862624 0.9950531 1.0796222 1.0758166 0.9933879 1.0309413 1.2412704
## [127] 0.9362770 0.9881374 1.0331969 1.0971090 0.9913557 2.1026850 1.0564535
## [134] 0.9437279 1.2086341 1.1265219 1.0885140 1.0245776 1.0331371 0.9827936
## [141] 0.9812230 0.9392177 1.0580851 0.9974539 1.0482408 0.9638605 1.0564212
## [148] 1.0026216 1.1550246 1.0606802
```

```
plot(lof.scores)
```




```
numero.de.outliers = 4

indices.de.lof.outliers.ordenados = order(lof.scores, decreasing = TRUE)
indices.de.lof.outliers.ordenados
```

```
##   [1]  42 118 132 110 107  16  61  23   9  86  33  60 109  63  24  34  37 119
##  [19]  32  44  38 115   7  14 126  94  99  39  58 135  21  88 120  73  57  25
##  [37] 149  77  36  54 136  19  69  15  85 114  51 116 130 101 137  17 122  90
##  [55] 123  71   6  12  66 150 103 102 143  48  27  22 133 147  82   5  89 145
##  [73]  68  97  43  81   4  49 106  52 112  95 129 139 125  55  53  98 138   2
##  [91]  50 117  11   3 100  70  64  59 113  87 111 104   1  26 148  30  72  45
## [109]  67 144  31 121  47 124 108 131  40 128  74   8  20  62 140  18  41  13
## [127] 141 105  46  10  35  78  96  83  79  28  75 146  84  92  65  93 134  80
## [145] 142  29  91 127  56  76
```

```
indices.de.lof.top.outliers = indices.de.lof.outliers.ordenados[1:4]
indices.de.lof.top.outliers
```

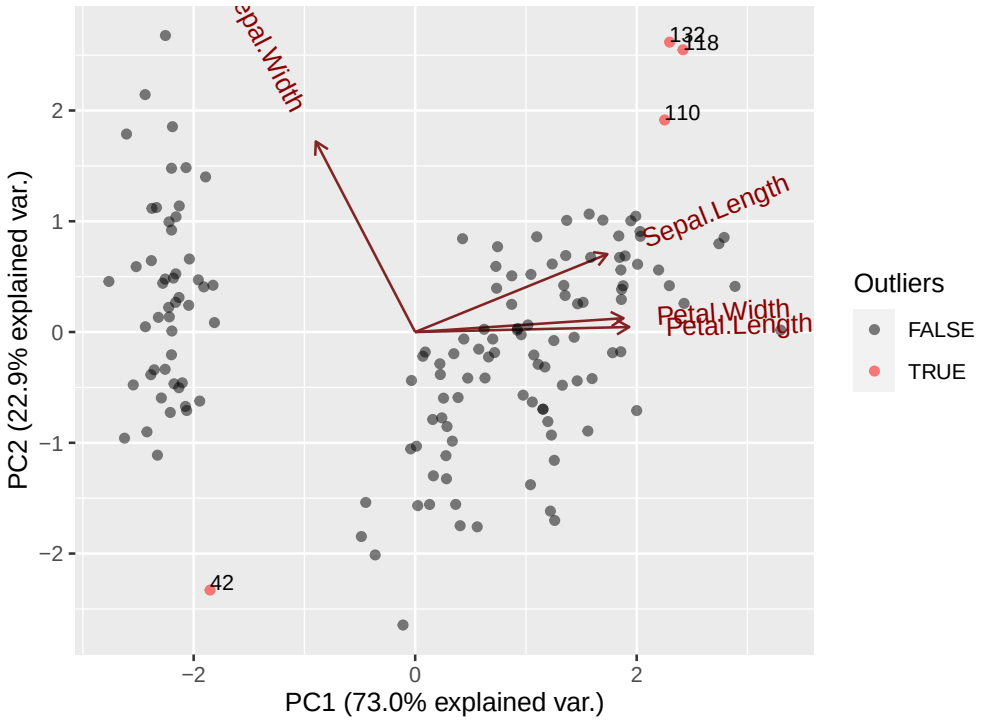
```
## [1]  42 118 132 110
```

```
is.lof.outlier = c(1:nrow(mis.datos.originales)) %in% indices.de.lof.top.outliers
is.lof.outlier
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE
```

```
MiBiPlot_Multivariate_Outliers(mis.datos.numericos, is.lof.outlier, "")
## MiBiPlot_Multivariate_Outliers
```

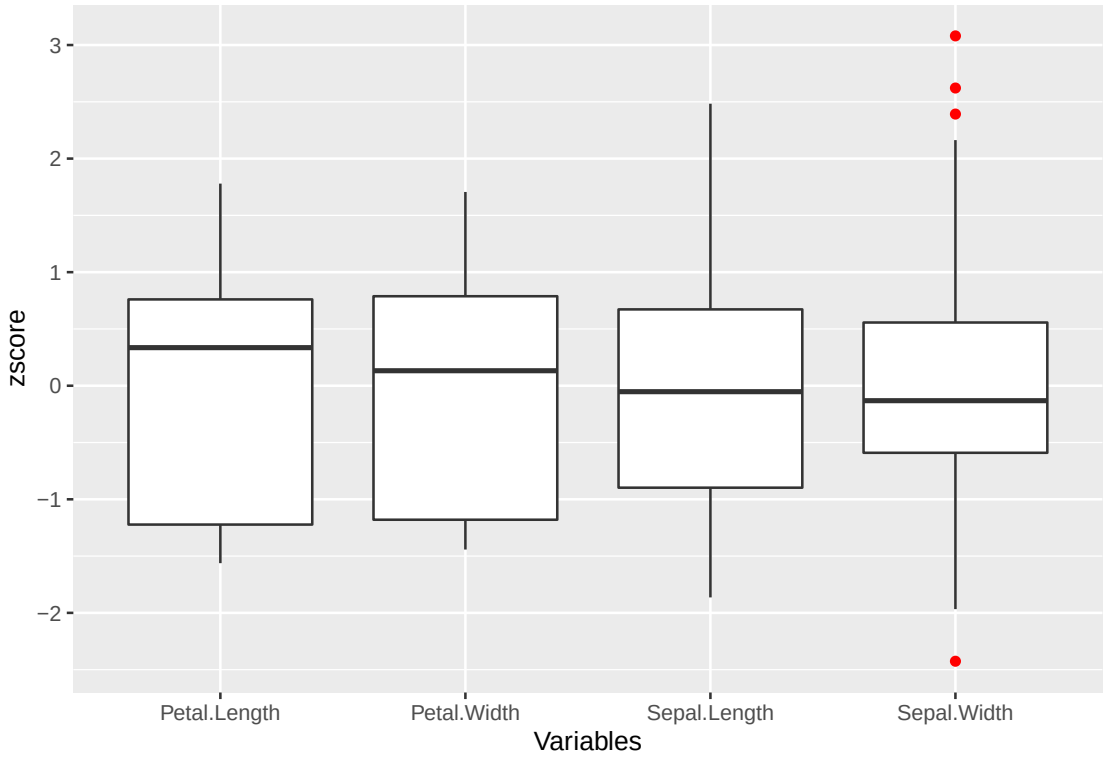



```
# Comparamos con los outliers en una sola dimensión que habríamos obtenido con
# el método IQR.
# Construimos las variables:

# vector.claves.outliers.IQR.en.alguna.columna: Contiene los índices de los que son
# outliers en alguna columna.
# Hay que llamar a la función vector_claves_outliers_IQR_en_alguna_columna
# vector.es.outlier.IQR.en.alguna.columna: Vector de T/F indicando si cada dato
# es outlier o no según el criterio IQR.
# Hay que llamar a la función vector_es_outlier_IQR_en_alguna_columna.

# Debe salir lo siguiente:
# vector.claves.outliers.IQR.en.alguna.columna
# [1] 16 33 34 61
```

```
# Mostramos el Biplot usando el vector de T/F vector.es.outlier.IQR.en.alguna.columma

# Construimos la variable
# indices.de.outliers.multivariantes.LOF.pero.no.1variantes: Contiene los outliers LOF
# que no son outliers IQR.
# Para ello, usamos setdiff y vemos que el resultado es el mismo conjunto de outliers LOF
# es decir, que ningún outlier LOF es outlier IQR.

# indices.de.outliers.multivariantes.LOF.pero.no.1variantes
# [1]  42 118 132 110

# COMPLETAR

vector.claves.outliers.IQR.en.alguna.columna =
  vector_claves_outliers_IQR_en_alguna_columna(mis.datos.numericos, coef = 1.5)
vector.claves.outliers.IQR.en.alguna.columna
```

```
## [1] 16 33 34 61
```

```
MiBoxPlot_juntos(mis.datos.numericos, vector.claves.outliers.IQR.en.alguna.columna)
```




```
indices.de.outliers.multivariantes.LOF = which(is.lof.outlier == TRUE)

indices.de.outliers.multivariantes.LOF.pero.no.1variantes = setdiff(
  indices.de.outliers.multivariantes.LOF, vector.claves.outliers.IQR.en.alguna.columna)

indices.de.outliers.multivariantes.LOF.pero.no.1variantes
```

```
## [1]  42 110 118 132
```

!MasterIA_GuionPracticas_Outliers_D2_ClusterBasedOutliers

```
##############################################################################
# MULTIVARIATE STATISTICAL OUTLIERS. CLUSTERING OUTLIERS
##############################################################################

# Los outliers son respecto a un conjunto de variables.

#####################################################################
# Lectura de valores y Preprocesamiento
#####################################################################

# Trabajamos sobre las columnas numéricas de iris [1:4].
# Este conjunto de datos está disponible en R.
# Tanto LOF como clustering usan distancias entre registros, por lo que habrá
# que trabajar sobre los datos previamente normalizados.

# Construimos los siguiente conjuntos:

# mis.datos.numericos -> con las columnas 1:4 de iris.
# mis.datos.numericos.normalizados -> con los valores normalizados.
# a Los rownames de mis.datos.numericos.normalizados les asignamos los rownames de
# mis.datos.numericos.

# Establecemos la variable numero.de.outliers a 5 y numero.de.clusters a 3.

mis.datos.numericos = iris[,1:4]
mis.datos.numericos.normalizados = scale(mis.datos.numericos)
rownames(mis.datos.numericos.normalizados) = rownames(mis.datos.numericos)

numero.de.outliers = 5
numero.de.clusters = 3

set.seed(2)  # Para establecer la semilla para la primera iteración de kmeans

#############################################################################
# Cómputo de los outliers según la distancia euclídea de cada dato
# al centroide de su cluster.
# El centroide podrá ser cualquiera (podrá provenir de un k-means
# o ser un medoide, por ejemplo).
#############################################################################

# k-Means

# Construimos el modelo kmeans (modelo.kmeans) con los datos normalizados.
# Para ello, usamos la función de R llamada "kmeans".

# A partir del resultado de kmeans, accedemos a:

# a) $cluster para obtener
```

```r
# los índices de asignación de cada dato al cluster correspondiente.
# El resultado lo guardamos en la variable indices.clustering.iris.
# Por ejemplo, si el dato con índice 69 está asignado al tercer cluster,
# en el vector indices.clustering.iris habrá un 3 en la componente número 69.

# b) $centers para obtener los datos de los centroides.
# Los datos están normalizados por lo que los centroides también lo están.
# El resultado lo guardamos en la variable centroides.normalizados.iris

# indices.clustering.iris
# 1   2   3   4   ... 69  70  71 ...
# 1   1   1   1   ... 3   3   2  ...

# centroides.normalizados.iris
#    Sepal.Length Sepal.Width Petal.Length Petal.Width
# 1  -1.01119138  0.85041372   -1.3006301  -1.2507035
# 2   1.13217737  0.08812645    0.9928284   1.0141287
# 3  -0.05005221 -0.88042696    0.3465767   0.2805873

# COMPLETAR

modelo.kmeans = kmeans(mis.datos.numericos.normalizados, numero.de.clusters)
indices.clustering.iris = modelo.kmeans$cluster
centroides.normalizados.iris = modelo.kmeans$centers

indices.clustering.iris
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   2   2   2   2   2   2   2   2   2   2   3   3   3   1   1   1   3   1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   1   3   1   1   1   1   3   1   1   1   1   3   3   3   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   1   1   1   3   3   1   1   1   1   1   1   1   1   1   1   1   1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   1   3   3   3   3   1   3   3   3   3   3   3   1   1   3   3   3   3   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   1   3   1   3   3   1   3   3   3   3   3   3   1   1   3   3   3   1   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   1   3   3   3   1   3   3   1
```

```r
centroides.normalizados.iris
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.05005221 -0.88042696    0.3465767   0.2805873
## 2  -1.01119138  0.85041372   -1.3006301  -1.2507035
## 3   1.13217737  0.08812645    0.9928284   1.0141287
```

```r
# Calculamos la distancia euclídea de cada dato a su centroide
# (con los valores normalizados).
# Para ello, usad la siguiente función:

distancias_a_centroides = function (datos.normalizados,
                                    indices.asignacion.clustering,
                                    datos.centroides.normalizados) {

  sqrt(rowSums((datos.normalizados -
                  datos.centroides.normalizados[indices.asignacion.clustering,]) ^ 2))
}

# dist.centroides.iris
# 1          2          3             ......
# 0.21224719 0.99271979 0.64980753    ......

# Ordenamos dichas distancias a través de la función order y obtenemos
# los índices correspondientes. Nos quedamos con los primeros
# (tantos como diga la variable numero.de.outliers).

# top.outliers.iris
# [1]  42  16 132 118  61

# COMPLETAR

dist.centroides.iris = distancias_a_centroides(mis.datos.numericos.normalizados,
                     indices.clustering.iris, centroides.normalizados.iris)

dist.centroides.iris.ordered = order(dist.centroides.iris, decreasing = TRUE)
top.outliers.iris = dist.centroides.iris.ordered[1:numero.de.outliers]

dist.centroides.iris
```

```
##          1          2          3          4          5          6          7
## 0.21224719 0.99271979 0.64980753 0.90043954 0.40081182 1.20750269 0.50077168
##          8          9         10         11         12         13         14
## 0.09101975 1.41699943 0.78729327 0.78735613 0.27525909 1.03152318 1.33036767
##         15         16         17         18         19         20         21
## 1.63318426 2.39097792 1.20345649 0.21546645 1.20582692 0.86416678 0.50233278
##         22         23         24         25         26         27         28
## 0.66603310 0.68428828 0.47785541 0.36224071 0.98693280 0.22607352 0.29373456
##         29         30         31         32         33         34         35
## 0.25276420 0.64802729 0.79870753 0.52134601 1.57132248 1.87025929 0.76601866
##         36         37         38         39         40         41         42
## 0.54713039 0.62868698 0.45829036 1.22957840 0.14532143 0.20194091 2.66163878
##         43         44         45         46         47         48         49
## 0.90623502 0.49913793 0.91852203 1.01605653 0.86663433 0.72034591 0.72082821
##         50         51         52         53         54         55         56
## 0.30194189 0.94969901 0.99020420 0.72419115 0.96617285 0.90820971 0.35602646
##         57         58         59         60         61         62         63
## 1.00314223 1.49711767 1.11259349 0.77661127 1.96536543 0.77271645 1.25521261
##         64         65         66         67         68         69         70
## 0.65934255 0.73586930 0.98451145 0.79917371 0.56648821 1.19587266 0.67654881
##         71         72         73         74         75         76         77
```

```
## 1.18403557 0.52883896 0.78969753 0.57179658 0.90287314 1.02741732 1.08939645
##         78         79         80         81         82         83         84
## 0.51876463 0.58777761 0.76176891 0.89438614 0.98220802 0.39082749 0.54042536
##         85         86         87         88         89         90         91
## 0.90265419 1.39386507 0.77749506 1.05774967 0.81590025 0.59787905 0.48984029
##         92         93         94         95         96         97         98
## 0.84127588 0.38811327 1.52759268 0.30728694 0.81453376 0.56239619 0.72429625
##         99        100        101        102        103        104        105
## 1.28375778 0.38036372 1.05492381 0.76330053 0.52296801 0.76978600 0.52854495
##        106        107        108        109        110        111        112
## 1.19820311 1.22141419 0.91967832 1.39845666 1.47828134 0.47715060 1.02915085
##        113        114        115        116        117        118        119
## 0.27726063 0.94542082 1.38967803 0.68393590 0.46307922 2.09425567 1.82481066
##        120        121        122        123        124        125        126
## 1.17397090 0.52364782 0.90906516 1.46551332 0.84452547 0.51835646 0.66437594
##        127        128        129        130        131        132        133
## 0.79507685 0.94608205 0.83817520 0.75495281 1.06767181 2.16620031 0.87389558
##        134        135        136        137        138        139        140
## 0.79374139 0.80252234 1.25562227 1.06856050 0.51262657 0.96693203 0.22973182
##        141        142        143        144        145        146        147
## 0.57181922 0.50960181 0.76330053 0.53961093 0.84844265 0.52300623 1.02749641
##        148        149        150
## 0.44223037 1.08075040 1.00173079
```

```
top.outliers.iris
```

```
## [1]  42  16 132 118  61
```