Trabajo Práctico

Cristian Rohr

31 de enero de 2018

UNIVARIATE STATISTICAL OUTLIERS -> IQR

Detección heurística de Outliers en 1 variable según el método IQR.

Voy a trabajar con el set de datos ***PimaIndiansDiabetes*** (https://data.world/uci/pima-indians-diabetes).

Nota: Dejo el codigo utilizado para cargar y preprocesar el dataset, para que se comprenda el contenido final del mismo.

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

Me quedo con las siguientes columnas: 2-7

Cada columna tiene el siguiente significado:

- Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)
- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function

Nota: Los valores codificados como 0 corresponden a missing data. Eliminare las observaciones con valores faltantes en alguna de sus columnas.

```
# Selecciono las columnas de interes
mydata.numeric  = mydata.numeric[, c(2:7)]

# Recodifico 0 como NA
is.na(mydata.numeric) <- !mydata.numeric
# Elimino filas con missing data
mydata.numeric <- mydata.numeric[complete.cases(mydata.numeric), ]

# Asigno nombre a las filas. Agrego el prefijo 'S'
nombres <- paste("S", seq(1:nrow(mydata.numeric)), sep = "")
rownames(mydata.numeric) <- nombres

# Columna sobre la que trabajare
indice.columna  = 4 # insulin
nombre.mydata   = "pima"
```

Preparación de datos

```
mydata.numeric.scaled = scale(mydata.numeric) # Normalizo
columna         = mydata.numeric[, indice.columna] # Me quedo con la columna de interés
nombre.columna  = names(mydata.numeric)[indice.columna] # Nombre de la columna de interés
columna.scaled  = mydata.numeric.scaled[, indice.columna] # Columna de interés normalizada

# Backups de datos
mydata.numeric.backup <- mydata.numeric
mydata.numeric.scaled.backup <- mydata.numeric.scaled
```

Parte primera. Cómputo de los outliers IQR

Calcular los outliers según la regla IQR. Directamente sin funciones propias

```
quantiles <- quantile(columna, c(0.25, 0.5, 0.75)) # Obtengo cuantiles
cuartil.primero <- quantiles[1] # Primer cuartil
cuartil.tercero <- quantiles[3] # Tercer cuartil
iqr <- cuartil.tercero - cuartil.primero # Rango intercuartil

# Valor de corte superior para outliers normales
extremo.superior.outlier.normal  = cuartil.tercero + 1.5*iqr
# Valor de corte inferior para outliers normales
extremo.inferior.outlier.normal  = cuartil.primero - 1.5*iqr
# Valor de corte superior para outliers extremos
extremo.superior.outlier.extremo = cuartil.tercero + 3*iqr
# Valor de corte inferior para outliers extremos
extremo.inferior.outlier.extremo = cuartil.primero - 3*iqr

vector.es.outlier.normal <- ifelse(columna < extremo.inferior.outlier.normal,
                                   TRUE,
                                   ifelse(columna > extremo.superior.outlier.normal,
                                          TRUE, FALSE))

print("Total de observacioness")
```

```
## [1] "Total de observacioness"
```

```
length(vector.es.outlier.normal) # Total de observaciones
```

```
## [1] 392
```

¿Cuantos outliers normales hay?

Hay 25 outliers normales.

¿Cuantos outliers extremos hay?

```
vector.es.outlier.extremo <- ifelse(columna < extremo.inferior.outlier.extremo,
                                    TRUE,
                                    ifelse(columna > extremo.superior.outlier.extremo,
                                           TRUE, FALSE))
```

Hay 8 outliers extremos.

Se calcularon los valores de corte para definir si una observación es un outlier o no. Definimos outliers normales como aquellos que estan por fuera del rango definido por: [1er cuartil - $1.5 IQR$, *3er cuartil +*

*1.5*IQR], donde IQR es el rango intercuartil.

Los outliers extremos son aquellos fuera del rango [1er cuartil - 3*IQR, 3er cuartil + 3*IQR].

Se crearon vectores booleanos que indican si cada observación es un outlier normal u extemo. Y se calculo el total de observaciones que son outliers normales y extremos.

Índices y valores de los outliers

Outliers normales

```
claves.outliers.normales <- which(vector.es.outlier.normal)
claves.outliers.normales
```

```
##  [1]   4   5  52  74  89 107 111 113 120 121 124 136 145 185 201 207 212
## [18] 250 302 330 338 358 365 368 388
```

Obtuvimos los índices del vector de aquellos valores que son outliers normales para la columna *insulin*.

```
data.frame.outliers.normales <- mydata.numeric[claves.outliers.normales, ]
data.frame.outliers.normales
```

```
##      glucose pressure triceps insulin mass pedigree
## S4       197       70      45     543 30.5    0.158
## S5       189       60      23     846 30.1    0.398
## S52      155       62      26     495 34.0    0.543
## S74      153       82      42     485 40.6    0.687
## S89      181       68      36     495 30.1    0.615
## S107     177       60      29     478 34.6    1.072
## S111     197       70      39     744 36.7    2.329
## S113     134       80      37     370 46.2    0.238
## S120     165       90      33     680 52.3    0.427
## S121     124       70      33     402 35.4    0.282
## S124     193       50      16     375 25.9    0.655
## S136     155       84      44     545 38.7    0.619
## S145     146       70      38     360 28.0    0.337
## S185     173       82      48     465 38.4    2.137
## S201     131       64      14     415 23.7    0.389
## S207     172       68      49     579 42.4    0.702
## S212     173       84      33     474 35.7    0.258
## S250     139       62      41     480 40.7    0.536
## S302     124       76      24     600 28.7    0.687
## S330     157       74      35     440 39.4    0.134
## S338     155       52      27     540 38.7    0.240
## S358     142       90      24     480 30.4    0.128
## S365     158       64      13     387 31.2    0.295
## S368     187       50      33     392 33.9    0.826
## S388     181       88      44     510 43.3    0.222
```

Mostramos un dataframe con las muestras que son outliers normales para la columna *insulin*.

```
nombres.outliers.normales <- row.names(data.frame.outliers.normales)
nombres.outliers.normales
```

```
##  [1] "S4"   "S5"   "S52"  "S74"  "S89"  "S107" "S111" "S113" "S120" "S121"
## [11] "S124" "S136" "S145" "S185" "S201" "S207" "S212" "S250" "S302" "S330"
## [21] "S338" "S358" "S365" "S368" "S388"
```

Mostramos el nombre de las muestras que son outliers normales para la columna *insulin*.

```
valores.outliers.normales <- mydata.numeric[vector.es.outlier.normal, nombre.columna]
valores.outliers.normales
```

```
##  [1] 543 846 495 485 495 478 744 370 680 402 375 545 360 465 415 579 474
## [18] 480 600 440 540 480 387 392 510
```

Mostramos los valores que son outliers normales para la columna *insulin*.

Con los comandos anteriores obtuve los índices (número de fila) de los outliers normales, un dataframe que contiene las filas que tienen los outliers normales, los nombres de las muestras con los outliers normales y los valores de los outliers normales.

Outliers extremos

```
claves.outliers.extremos <- which(vector.es.outlier.extremo)
claves.outliers.extremos
```

```
## [1]   4   5 111 120 136 207 302 338
```

```
data.frame.outliers.extremos <- mydata.numeric[vector.es.outlier.extremo, ]
data.frame.outliers.extremos
```

```
##      glucose pressure triceps insulin mass pedigree
## S4       197       70      45     543 30.5    0.158
## S5       189       60      23     846 30.1    0.398
## S111     197       70      39     744 36.7    2.329
## S120     165       90      33     680 52.3    0.427
## S136     155       84      44     545 38.7    0.619
## S207     172       68      49     579 42.4    0.702
## S302     124       76      24     600 28.7    0.687
## S338     155       52      27     540 38.7    0.240
```

```
nombres.outliers.extremos <- row.names(data.frame.outliers.extremos)
nombres.outliers.extremos
```

```
## [1] "S4"   "S5"   "S111" "S120" "S136" "S207" "S302" "S338"
```

```
valores.outliers.extremos <- mydata.numeric[vector.es.outlier.extremo, nombre.columna]
valores.outliers.extremos
```

```
## [1] 543 846 744 680 545 579 600 540
```

Con los comandos anteriores obtuve los índices (número de fila) de los outliers extremos, un dataframe que contiene las filas que tienen los outliers extremos, los nombres de las muestras con los outliers extremos y los valores de los outliers extremos.

Desviación de los outliers con respecto a la media de la columna

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

```
##       S4       S5      S52      S74      S89     S107     S111     S113
## 3.255961 5.805571 2.852062 2.767917 2.852062 2.709015 4.947286 1.800243
##     S120     S121     S124     S136     S145     S185     S201     S207
## 4.408755 2.069508 1.842315 3.272790 1.716097 2.599625 2.178898 3.558885
##     S212     S250     S302     S330     S338     S358     S365     S368
## 2.675356 2.725844 3.735590 2.389262 3.230717 2.725844 1.943290 1.985363
##     S388
```

```
## 2.978280
```

Me fijo cuanto se desvian los outliers con respecto a la media de la columna. Recordar que los datos fueron normalizados con un Z-score.

Repito paralos outliers extremos

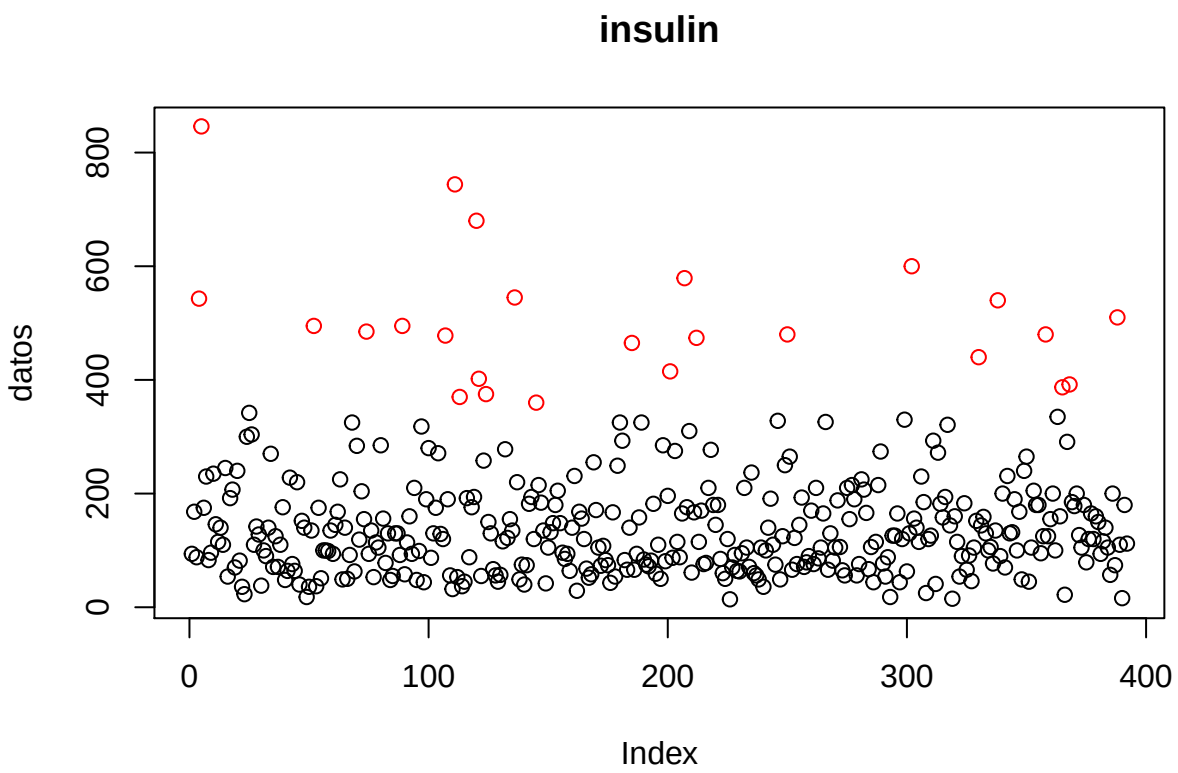
```
valores.normalizados.outliers.extremos <- columna.scaled[vector.es.outlier.extremo]
valores.normalizados.outliers.extremos
```

```
##       S4       S5     S111     S120     S136     S207     S302     S338
## 3.255961 5.805571 4.947286 4.408755 3.272790 3.558885 3.735590 3.230717
```

Plot

```
# Primero lo hago con los datos sin normalizar
# outliers normales
MiPlot_Univariate_Outliers(columna, claves.outliers.normales, nombre.columna)
```

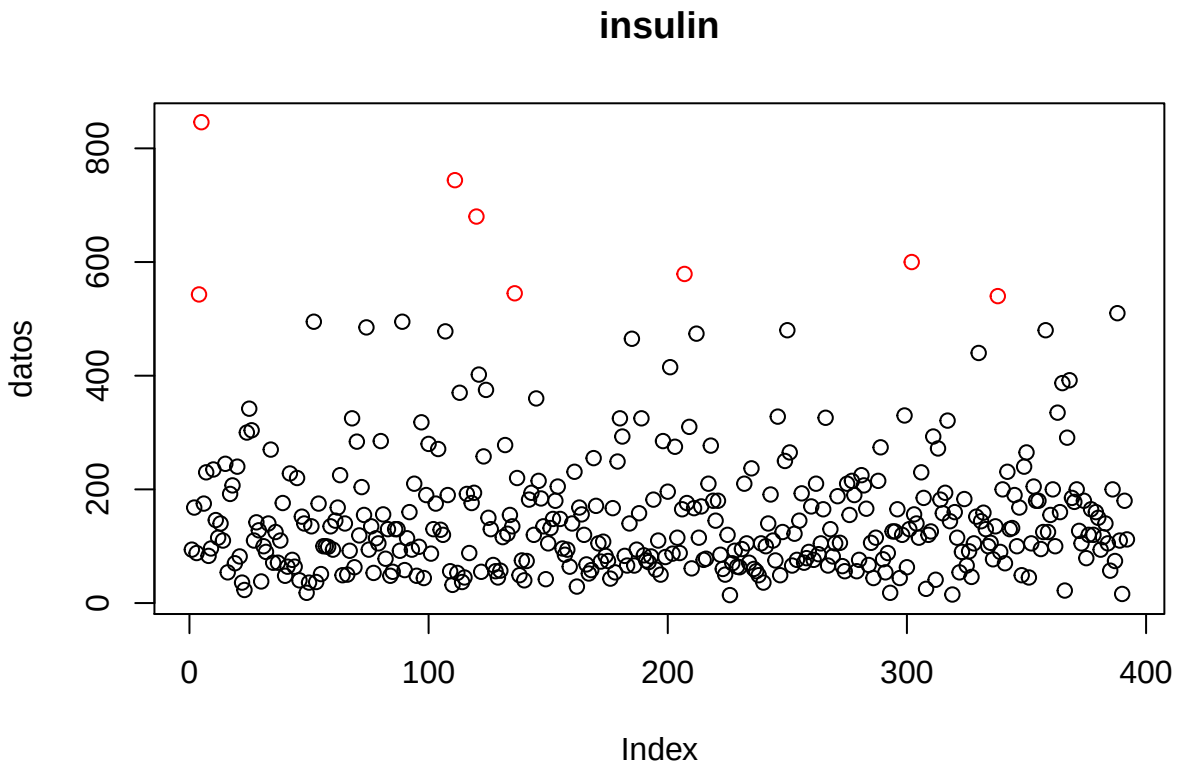
```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```




```
# outliers extremos
MiPlot_Univariate_Outliers(columna, claves.outliers.extremos, nombre.columna)
```

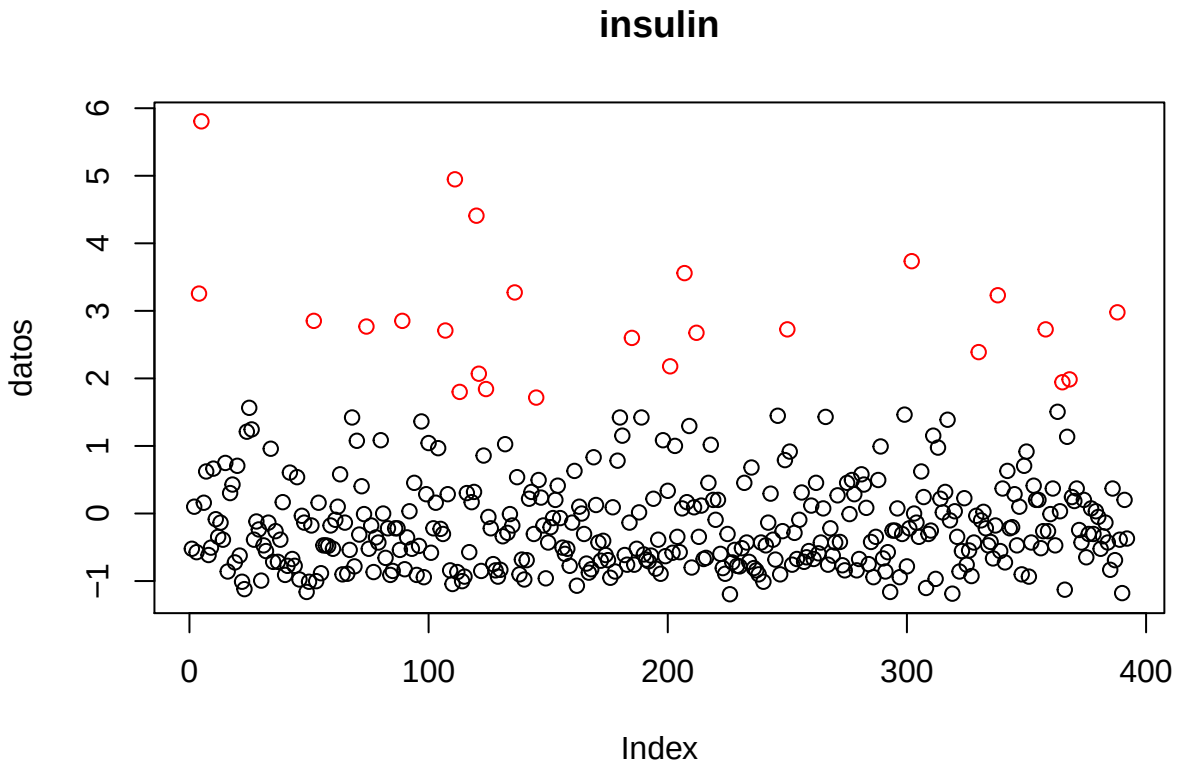
```
##
## N?mero de datos: 392
```

```
## ?Qui?n es outlier?: FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```




```
# Repito con los datos normalizados
# Outliers normales
MiPlot_Univariate_Outliers(columna.scaled, claves.outliers.normales, nombre.columna)
```

```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```




```
# Outliers extremos
MiPlot_Univariate_Outliers(columna.scaled, claves.outliers.extremos, nombre.columna)
```

```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

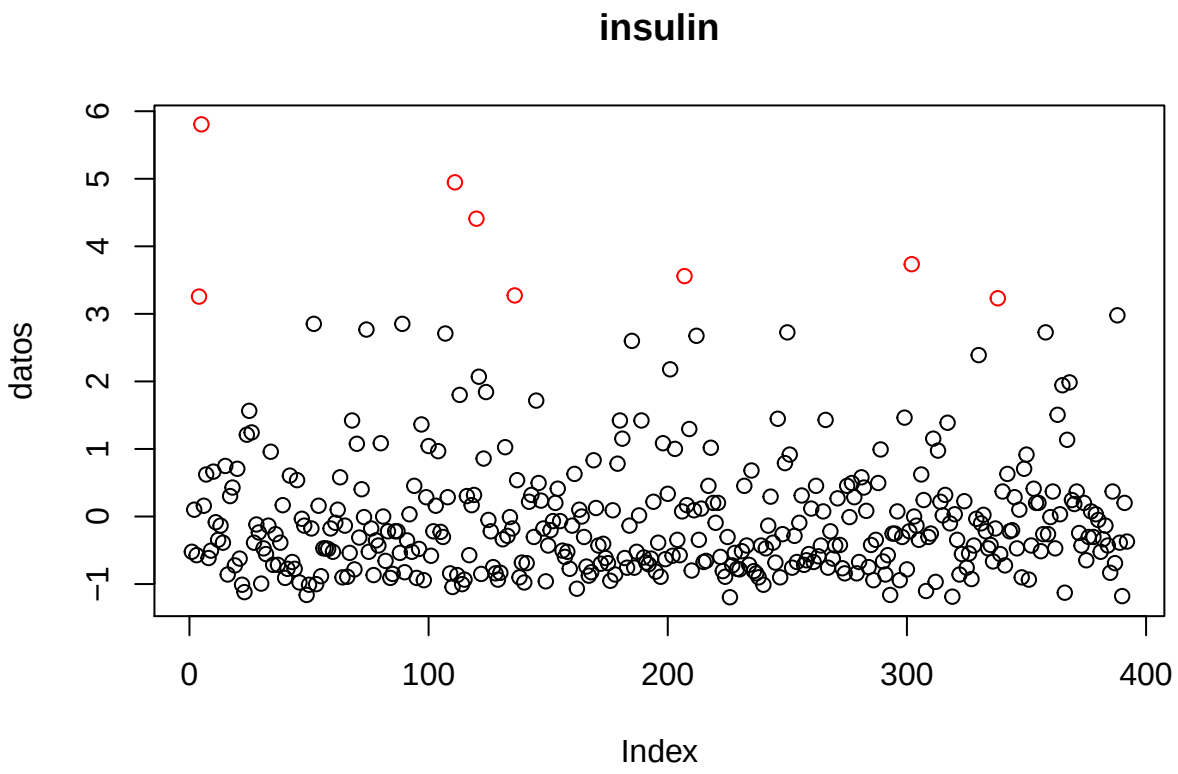
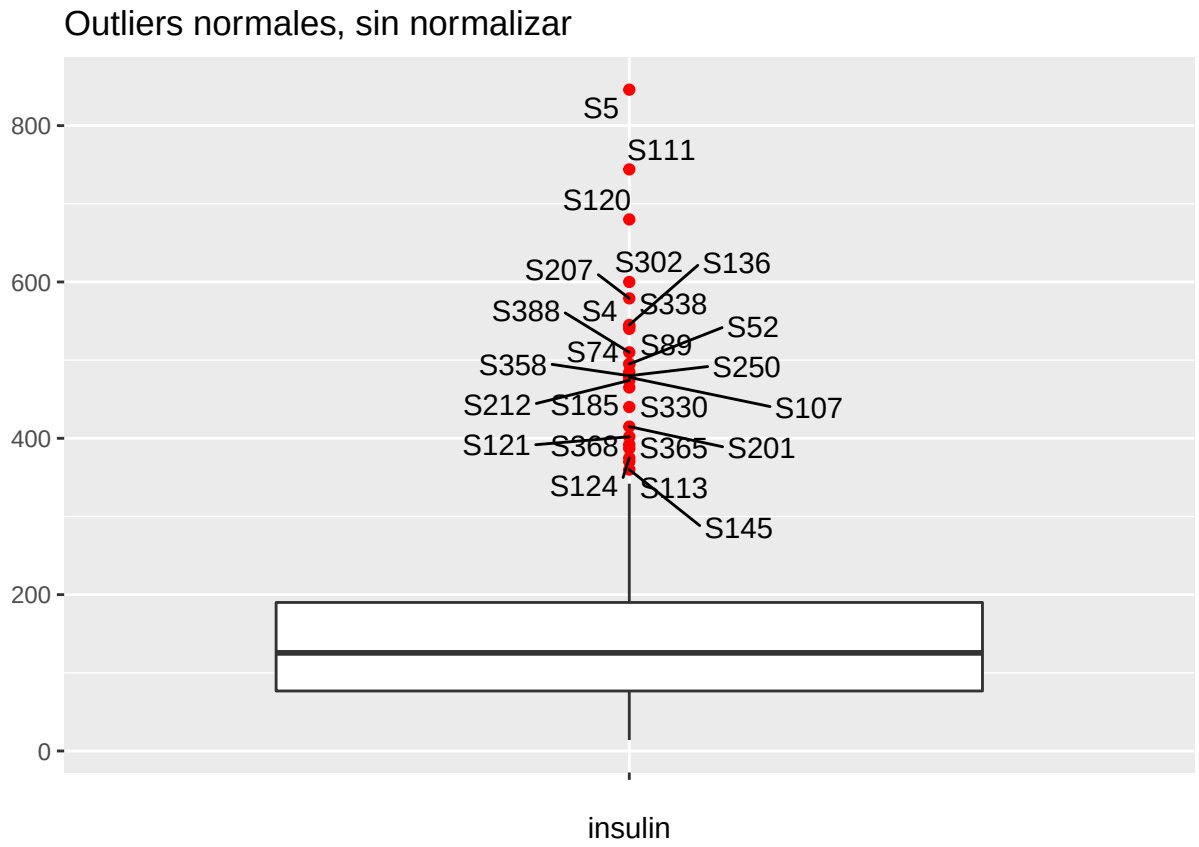


Gráfico de puntos, de los índices vs el valor. Se pueden observar a los outliers marcados en color rojo (tanto extremos como normales, dependiendo del gráfico).

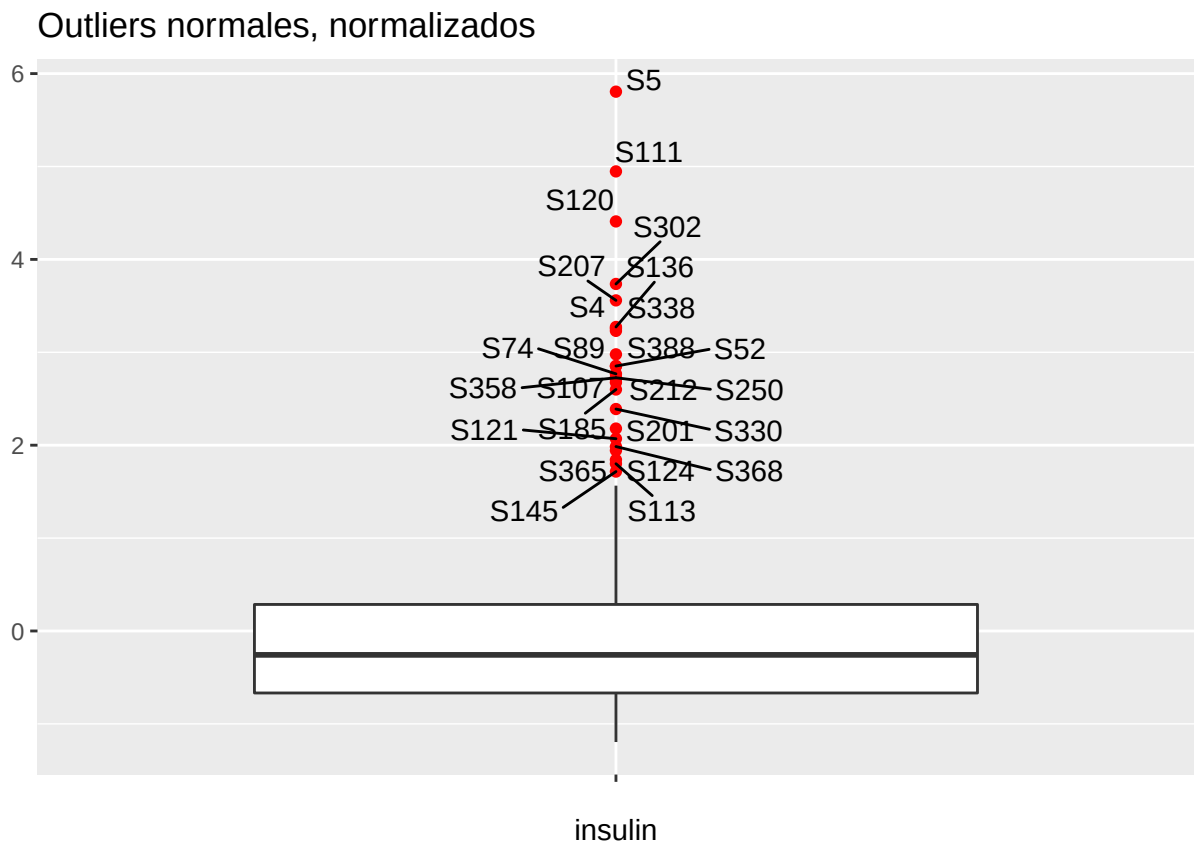
Al observar el gráfico normalizado y no normalizado son iguales, cambia la escala del eje Y, pero no la distancia entre los datos. La normalización no afecta la posición de los datos.

Boxplot

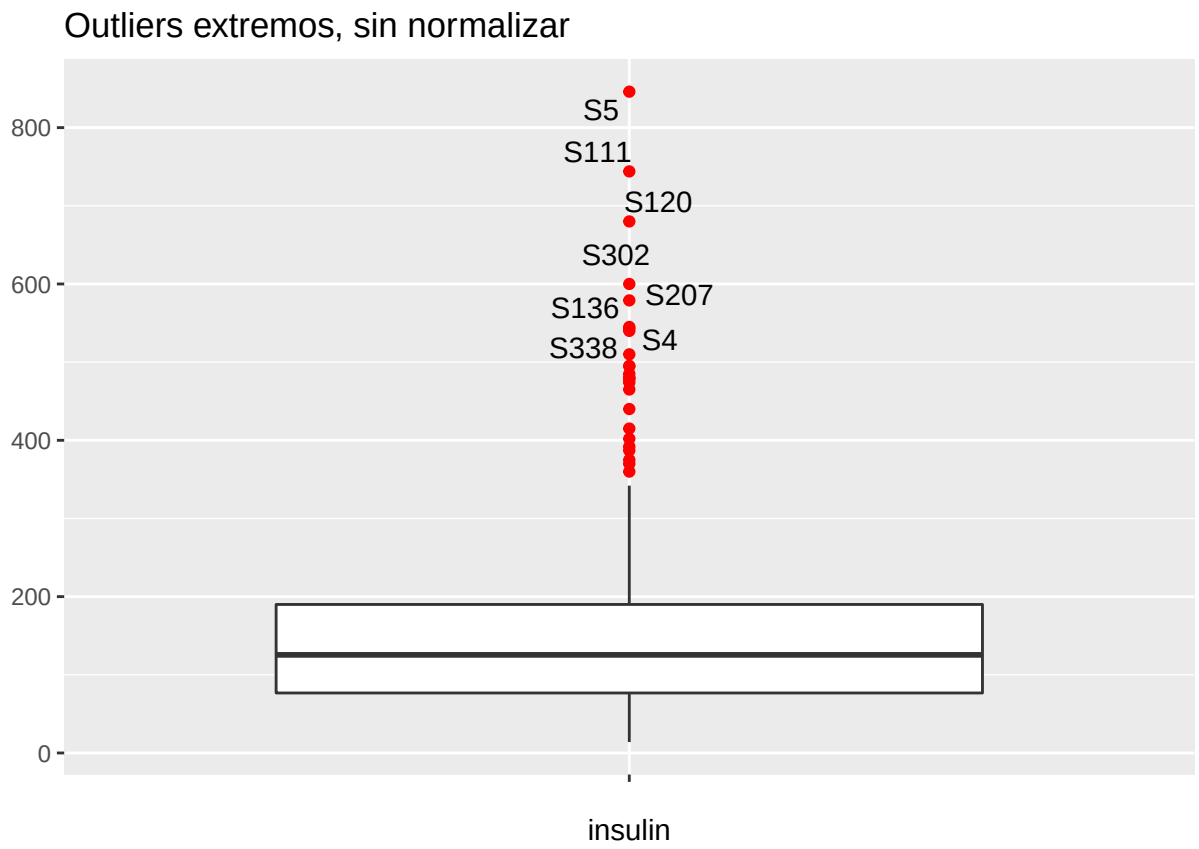
```
# Outliers normales
# Datos sin normalizar
plot.out.normales.sinnormalizar <- MiBoxPlot_IQR_Univariate_Outliers(datos = mydata.numeric,
                                  indice.de.columna = indice.columna,
                                  coef = 1.5)
plot.out.normales.sinnormalizar <- plot.out.normales.sinnormalizar +
  ggtitle("Outliers normales, sin normalizar")
plot.out.normales.sinnormalizar
```



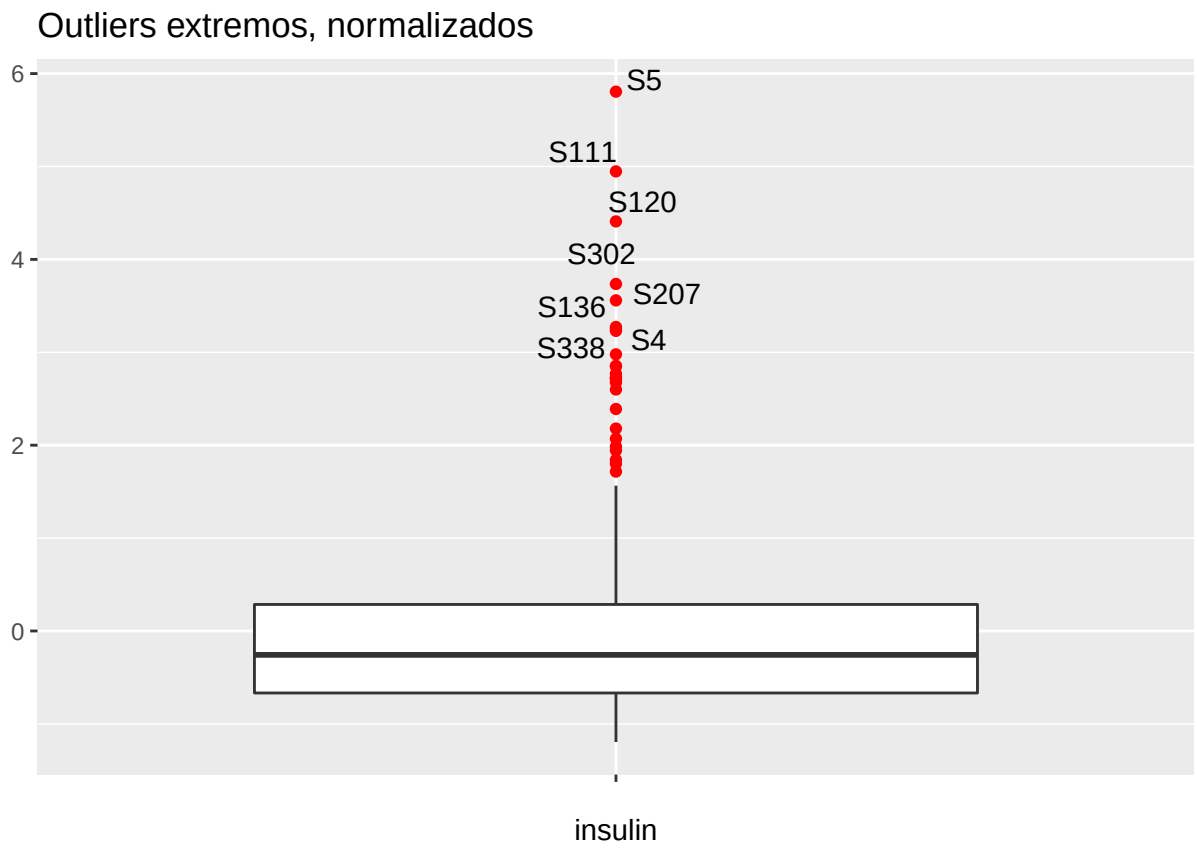

```
# Datos Normalizados
plot.out.normales.normalizados <- MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric.scaled,
                                  indice.columna,
                                  coef = 1.5)

plot.out.normales.normalizados <- plot.out.normales.normalizados +
  ggtitle("Outliers normales, normalizados")
plot.out.normales.normalizados
```




```
# Outliers extremos
# Para ver los outliers extremos, debo setear el coeficiente en 3
# Datos sin normalizar
plot.out.extremos.sinnormalizar <- MiBoxPlot_IQR_Univariate_Outliers(datos = mydata.numeric,
                                   indice.de.columna = indice.columna,
                                   coef = 3)
plot.out.extremos.sinnormalizar <- plot.out.extremos.sinnormalizar +
  ggtitle("Outliers extremos, sin normalizar")
plot.out.extremos.sinnormalizar
```




```
# Datos normalizados
plot.out.extremos.normalizados <- MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric.scaled,
                                  indice.columna,
                                  coef = 3)
plot.out.extremos.normalizados <- plot.out.extremos.normalizados +
  ggtitle("Outliers extremos, normalizados")
plot.out.extremos.normalizados
```



Gráficos de caja para outliers normales (datos sin normalizar y normalizados) y para outliers extremos (datos normalizados y sin normalizar).

Cuando comparo los datos normalizados y no normalizados, vemos que el Boxplot es el mismo ya que la normalización no afecta a la posición relativa de los datos. Se pueden observar como los outliers estan por fuera de la caja que marca el comienzo y fin del 1er y 3er cuartil, y de la línea solida que continua, que marca el punto a partir del cual nos alejamos 1.5 veces del rango intercuartil

Nota: Hize una modificación a la función entregada en clase, para usar geom_text_repel del paquete ggrepel en lugar de geom_text, para evitar la superposición de las etiquetas.

Cómputo de los outliers IQR con funciones propias

```
# Outliers normales
print("Vector booleano que indica si un valor es outlier normal")
```

```
## [1] "Vector booleano que indica si un valor es outlier normal"
```

```
vector_es_outlier_IQR(datos = mydata.numeric,
                      indice.de.columna = indice.columna,
                      coef = 1.5)
```

```
##   [1] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [111]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [122] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [144] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [188] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [210] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [221] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [232] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [243] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [254] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [276] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [287] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [298] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [309] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [320] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [331] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [342] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [353] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [364] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [375] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [386] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
print("\n")
```

```
## [1] "\n"
```

```
print("Índices de los outliers normales")
```

```
## [1] "Índices de los outliers normales"
```

```
# Outliers normales
vector_claves_outliers_IQR(mydata.numeric, indice.columna, coef = 1.5)
```

```
##  [1]   4   5  52  74  89 107 111 113 120 121 124 136 145 185 201 207 212
## [18] 250 302 330 338 358 365 368 388
```

```
print("\n")
```

```
## [1] "\n"
```

```
print("Vector booleano que indica si un valor es outlier extremo")
```

```
## [1] "Vector booleano que indica si un valor es outlier extremo"
```

```
# Outliers extremos
vector_es_outlier_IQR(mydata.numeric, indice.columna, coef = 3)
```

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

```
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [111]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [188] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [210] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [221] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [232] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [243] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [254] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [276] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [287] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [298] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [309] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [320] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [331] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [342] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [353] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [364] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [375] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [386] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
print("\n")
```

```
## [1] "\n"
```

```
print("Índices de los outliers extremos")
```

```
## [1] "Índices de los outliers extremos"
```

```
# Outliers extremos
vector_claves_outliers_IQR(mydata.numeric, indice.columna, coef = 3)
```

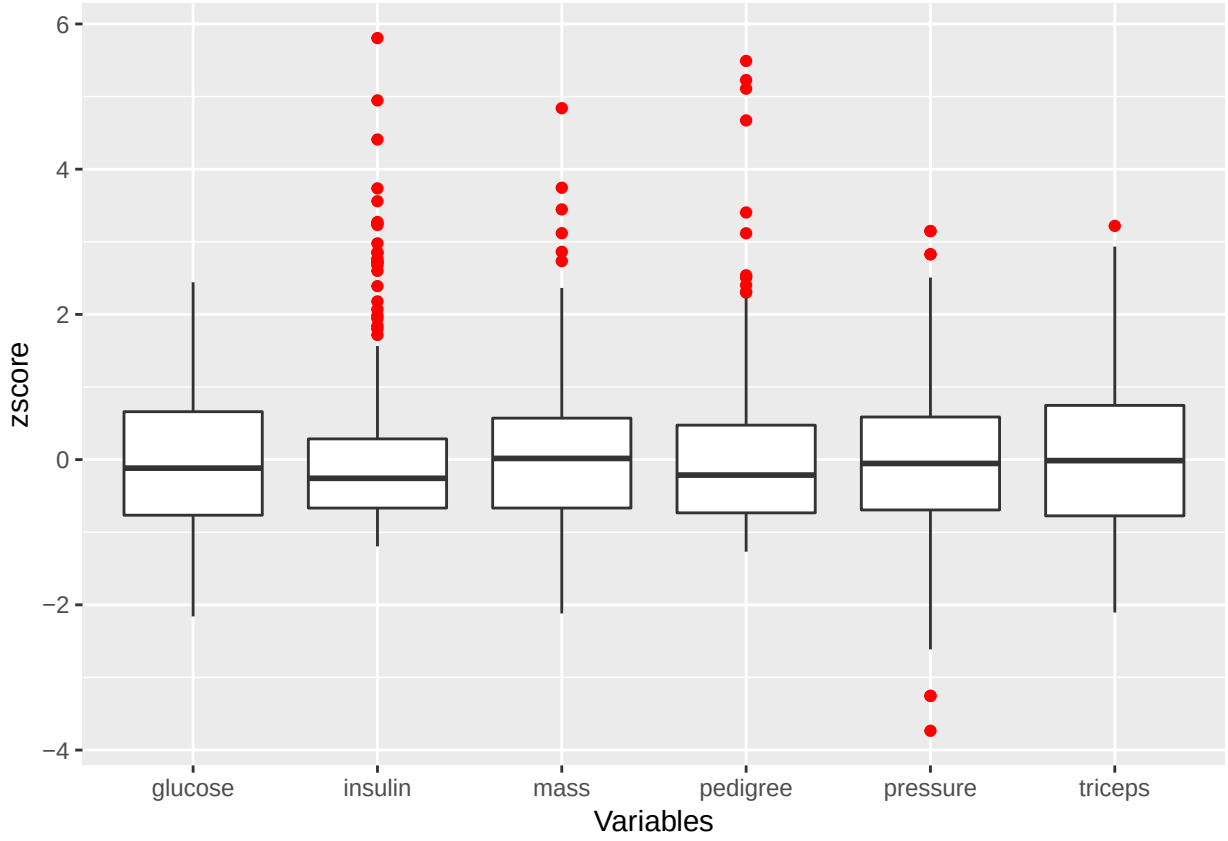
```
## [1]   4   5 111 120 136 207 302 338
```

La función vector_es_outlier_IQR devuelve un vector de booleanos indicando si un dato es outlier o no, en una determinada columna de un dataframe, de acuerdo a un coeficiente aplicado al rango intercuartil. Si el parámetro **coef** es 1.5 calculamos los llamados outliers normales, si es 3 los llamados outliers extremos.

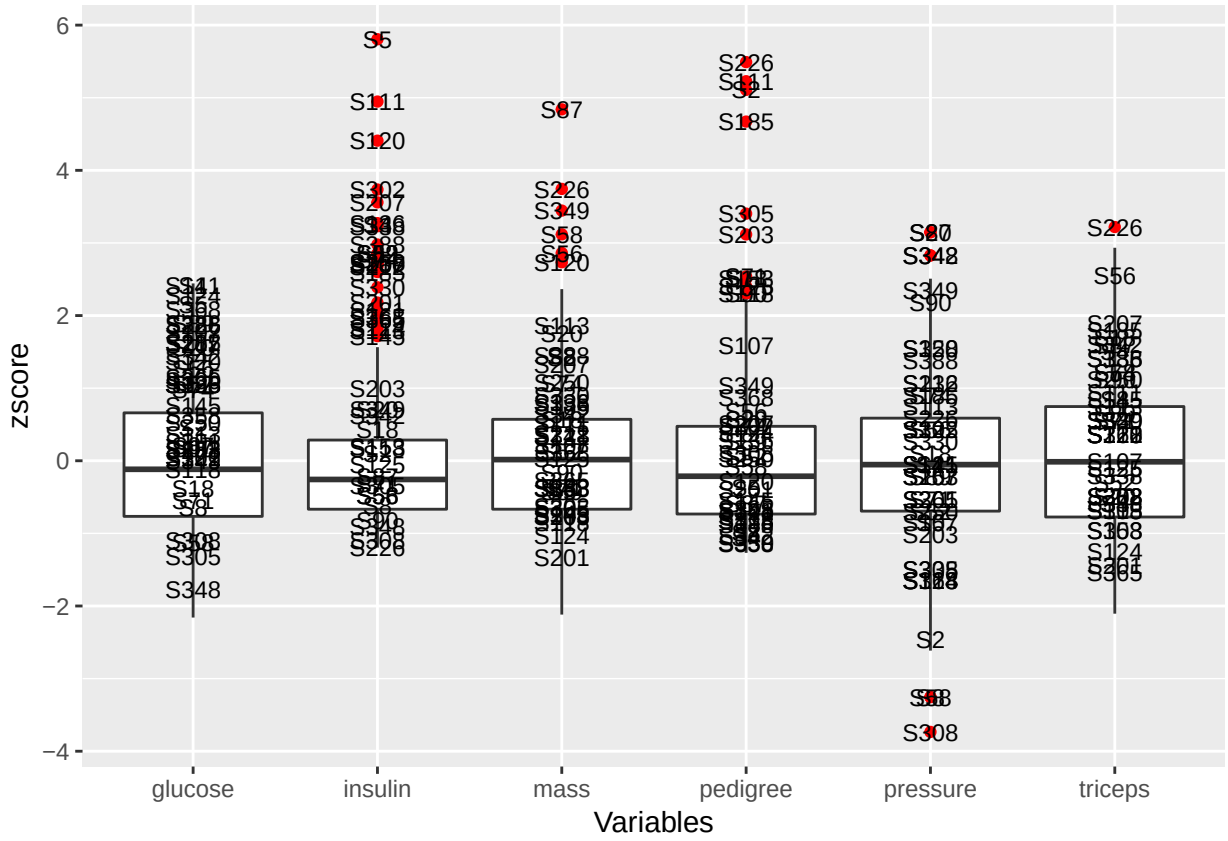
La función vector_claves_outliers_IQR retorna los índices de los outliers, nuevamente depende del parámetro **coef** calcularemos los outliers normales o extremos.

BoxPlot 2

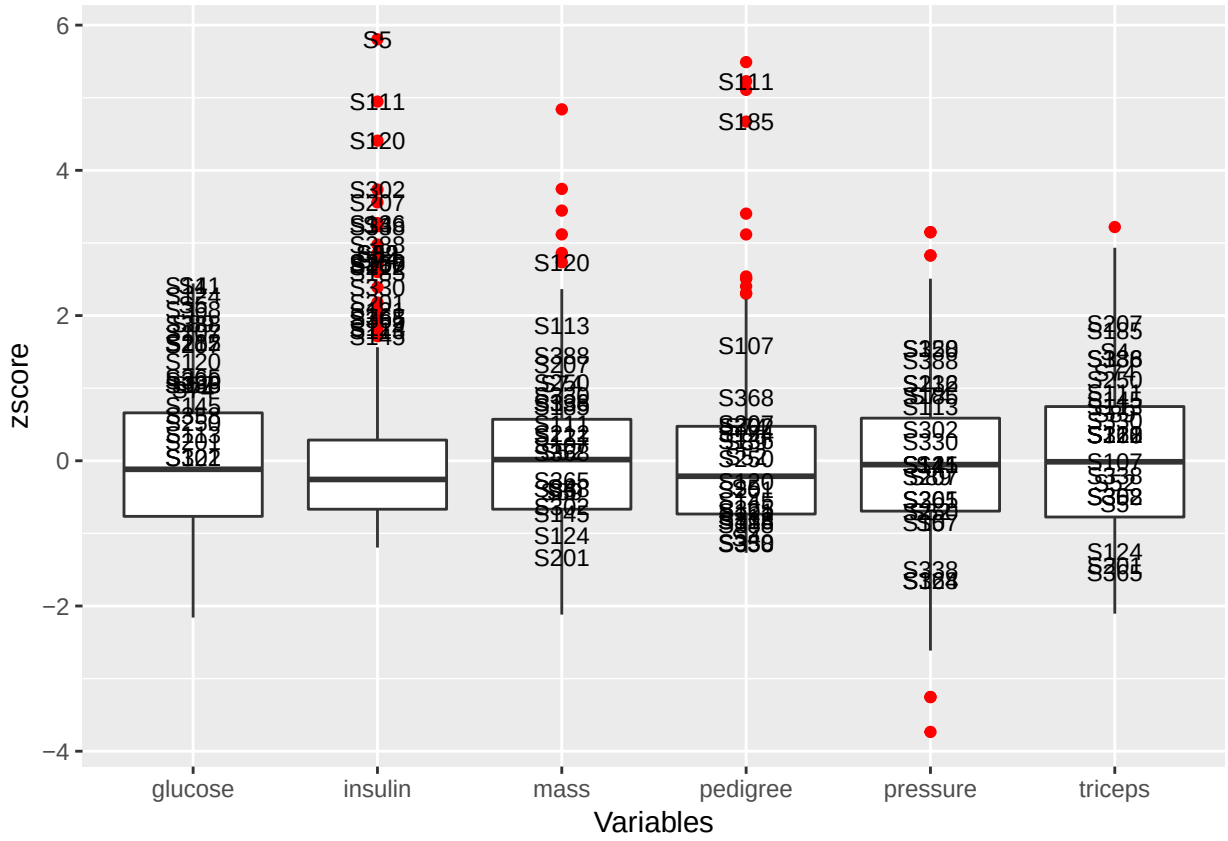
```
# Outliers normales
MiBoxPlot_juntos(mydata.numeric)
```




```
MiBoxPlot_juntos_con_etiquetas(mydata.numeric)
```




```
MiBoxPlot_juntos(mydata.numeric, vector.es.outlier.normal)
```



Nota: la función MiBoxPlot_juntos etiqueta los outliers, si se le pasa el vector booleano de outliers

Todos los outliers para *Insulin* no son outliers para otras variables como *glucose*, *pressure* o *triceps*.

Podemos observar que la muestra *S120* es un outlier extremo para *Insulin* y tambien es un outlier para *mass*.

La muestra *S111* es un outlier extremo para *Insulin* y tambien es un outlier para *pedigree*, todo parece indicar que también es un outlier extremo para dicha variable.

Trabajo sobre varias columnas

Los outliers seguirá siendo univariados, pero trabajo sobre varias columnas al mismo tiempo.

```
indices.de.outliers.en.alguna.columna <- unlist(sapply(1:ncol(mydata.numeric),
                                                       vector_claves_outliers_IQR,
                                                       datos = mydata.numeric,
                                                       coef = 1.5 )
                                                )
# Índices de outliers en alguna columna
indices.de.outliers.en.alguna.columna
```

```
##  [1]   8  20  58  87 308 342 348 226   4   5  52  74  89 107 111 113 120
## [18] 121 124 136 145 185 201 207 212 250 302 330 338 358 365 368 388  56
## [35]  58  87 120 226 349   2  18  71  90 111 118 125 153 185 203 226 305
```

```
# Obtenemos las filas únicas con outliers en al menos 1 columna
length(unique(indices.de.outliers.en.alguna.columna))
```

```
## [1] 44
```

Hay 44 filas con outliers en al menos una de las columnas

Chequeo la cantidad de outliers extremos en alguna columna

```
indices.de.outliers.extremos.en.alguna.columna <- unlist(sapply(1:ncol(mydata.numeric),
                                                                vector_claves_outliers_IQR,
                                                                datos = mydata.numeric,
                                                                coef = 3 )
)

# Índices de outliers extremos en alguna columna
indices.de.outliers.extremos.en.alguna.columna
```

```
##  [1]   4   5 111 120 136 207 302 338  87   2 111 185 226
```

```
# Obtenemos las filas únicas con outliers en al menos 1 columna
length(unique(indices.de.outliers.extremos.en.alguna.columna))
```

```
## [1] 12
```

En total 12 filas tienen outliers extremos en almenos 1 de las columnas

```
mydata.numeric.scaled[indices.de.outliers.extremos.en.alguna.columna, ]
```

```
##         glucose    pressure    triceps    insulin       mass   pedigree
## S4   2.40993414 -0.05307782  1.5076030  3.2559608 -0.3680065 -1.0566094
## S5   2.15070545 -0.85332804 -0.5843629  5.8055711 -0.4249245 -0.3619399
## S111 2.40993414 -0.05307782  0.9370668  4.9472864  0.5142218  5.2272550
## S120 1.37301935  1.54742262  0.3665307  4.4087549  2.7340221 -0.2780007
## S136 1.04898348  1.06727248  1.4125136  3.2727899  0.7988116  0.2777349
## S207 1.59984446 -0.21312786  1.8879604  3.5588848  1.3253027  0.5179747
## S302 0.04447227  0.42707231 -0.4892736  3.7355904 -0.6241373  0.4745579
## S338 1.04898348 -1.49352821 -0.2040055  3.2307171  0.7988116 -0.8192640
## S87  0.20649021  3.14792305  1.6026923 -0.2192507  4.8399865 -0.5906020
## S2   0.46571891 -2.45382847  0.5567094  0.1005024  1.4249091  5.1085822
## S111 2.40993414 -0.05307782  0.9370668  4.9472864  0.5142218  5.2272550
## S185 1.63224805  0.90722244  1.7928710  2.5996254  0.7561231  4.6715194
## S226 1.85907316  0.58712235  3.2192114 -1.1953391  3.7443158  5.4906505
```

Arriba vemos el dataframe de valores normalizados, de las muestras que son outlier extremo en alguna de las columnas. Podemos observar el caso de la muestra *S87* que se dispara mucho tanto en la columna *mass* como *pressure*. O el de la *S111* que se dispara en *pedigree* e *insulin*.

Nota: no mostre los outliers normales en alguna columna, para no generar un dataframe de 44 filas

Construyo una función que devuelve un vector de índices de aquellos registros que tienen un outlier en alguna de las columnas

```
vector_claves_outliers_IQR_en_alguna_columna <- function(datos, coef = 1.5) {
  indices <- unlist(sapply(1:ncol(datos),
    vector_claves_outliers_IQR,
    datos = datos,
    coef = 1.5 )
  )
  indices
}
```

```r
indices.de.outliers.en.alguna.columna.funcion <- vector_claves_outliers_IQR_en_alguna_columna(mydata.nu

# Chequeo que coincidan los valores
if(sum(indices.de.outliers.en.alguna.columna == indices.de.outliers.en.alguna.columna.funcion))
{
  print("Coincidencia. Función bien implementada.")
} else {
  print("Hay algo mal. Chequear la implementación de la función")
}
```

```
## [1] "Coincidencia. Función bien implementada."
```

La función esta implementada correctamente

Función que devuelve un vector booleano indicando si una muestra en outlier en alguna columna

```r
vector_es_outlier_IQR_en_alguna_columna = function(datos, coef = 1.5){
  indices.de.outliers.en.alguna.columna =  vector_claves_outliers_IQR_en_alguna_columna(datos, coef)
  todos = c(1:nrow(datos))
  bools = todos %in% indices.de.outliers.en.alguna.columna
  return (bools)
}

outliers.en.alguna.columna.funcion <- vector_es_outlier_IQR_en_alguna_columna(mydata.numeric)
outliers.en.alguna.columna.funcion
```

```
##   [1] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [56]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [89]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [111]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [122] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [144] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [188] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
## [210] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [221] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [232] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [243] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [254] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [276] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [287] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [298] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE
## [309] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [320] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [331] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [342]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
## [353] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [364] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [375] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [386] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

UNIVARIATE STATISTICAL OUTLIERS -> 1-variate Normal Distribution

Detección de Outliers en 1 variable normal aplicando tests estadísticos paramétricos.

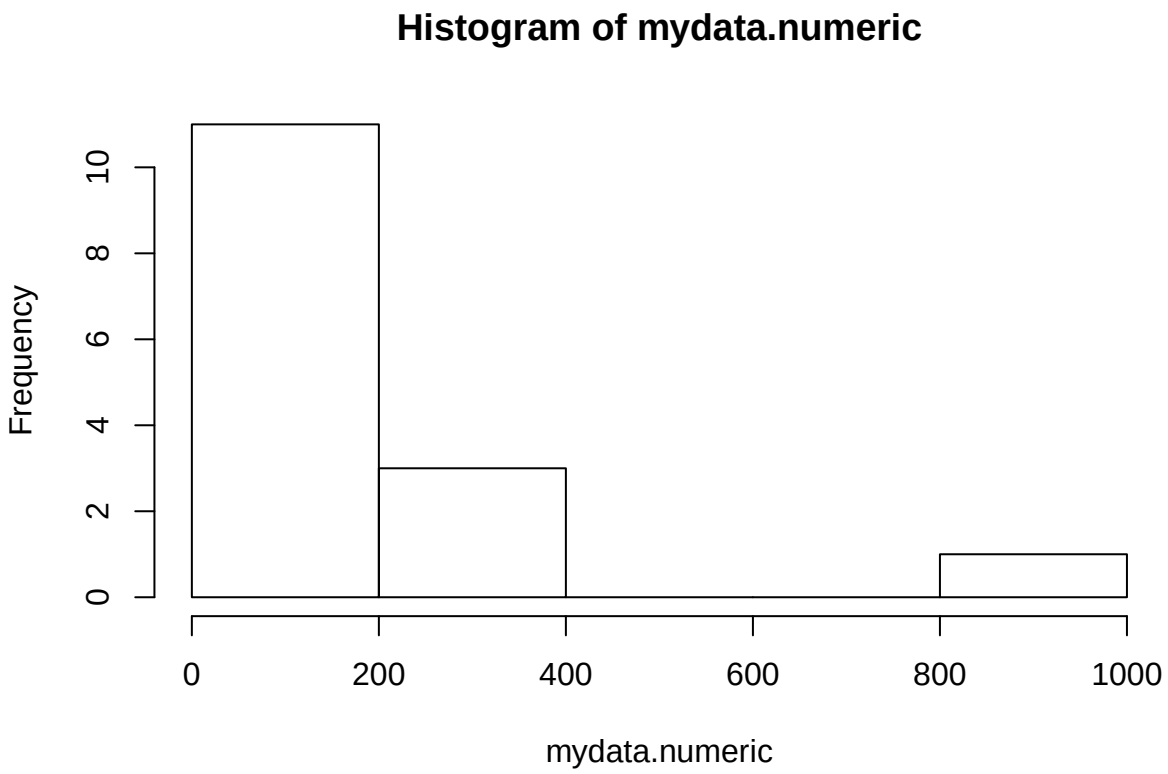
Genero datos con un outlier, para esto voy a tomar los valores de la columna Insulin del dataset pima, voy a quedarme con el outlier mas extremo, y remover el resto de outlier calculados en los puntos anteriores, además de downsamplear los datos.

Nota: Dejo el codigo utilizado para que se comprenda lo que realize.

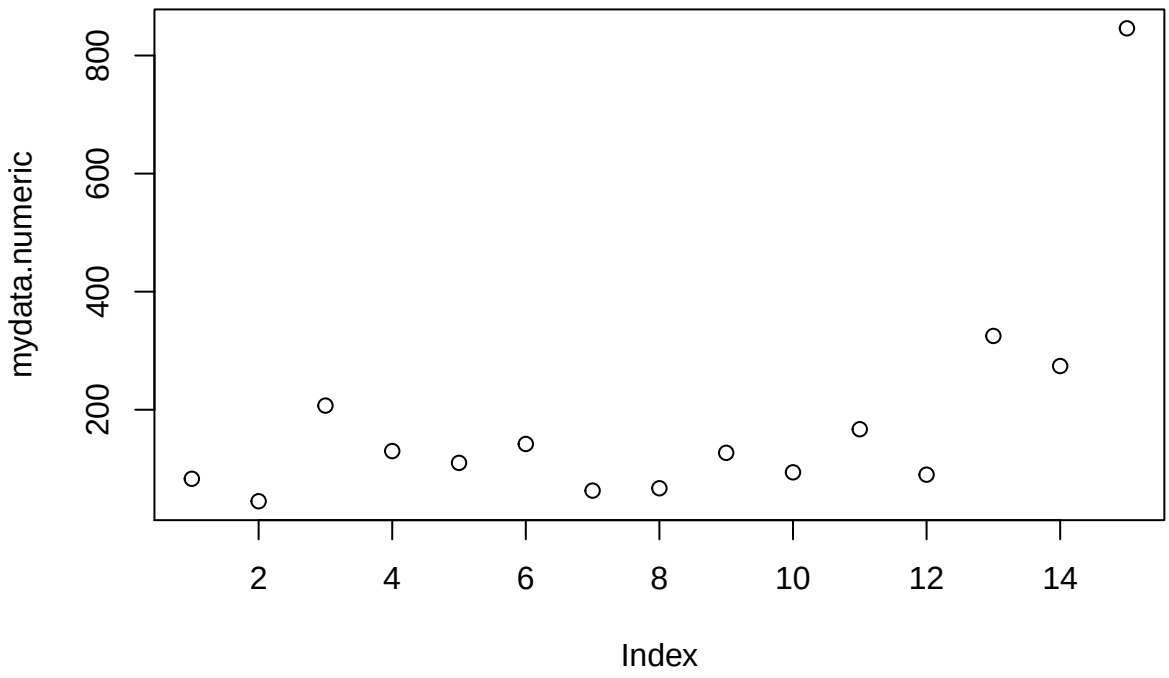
```
aux <- columna # Columna tiene los valores de Insulin
maximo <- max(columna) # Me quedo con el maximo

# El vector.es.outlier.normal es un vector TRUE/FALSE con los datos de que valor es outlier
vector.es.outlier.normal.aux <- vector.es.outlier.normal
names(vector.es.outlier.normal.aux) <- rownames(mydata.numeric.backup)

# Elimino los outliers
aux <- aux[!vector.es.outlier.normal.aux]

# Hago un downsample a 14 elementos
set.seed(87)
aux <- sample(x = aux, size = 14, replace = F)

# Agrego el valor outlier maximo
aux <- c(aux, maximo)

# Ahora si tengo mis datos para trabajar
datos.con.un.outlier <- aux
mydata.numeric = datos.con.un.outlier
max(mydata.numeric)
```

```
## [1] 846
```

```
hist(mydata.numeric)
```




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



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, que es el outlier mas extremo que deje intencionalmente.

```
test.de.Grubbs <- grubbs.test(mydata.numeric, two.sided = TRUE)
test.de.Grubbs
```

```
##
## Results of Hypothesis Test
## --------------------------
##
## Alternative Hypothesis:          highest value 846 is an outlier
##
## Test Name:                       Grubbs test for one outlier
##
## Data:                            mydata.numeric
##
## Test Statistics:                 G = 3.320842
##                                  U = 0.156021
##
## P-value:                         1.995367e-05
```

```
test.de.Grubbs$p.value
```

```
## [1] 1.995367e-05
```

El test de Grubb's es un test estadístico usado para detectar outliers en un data set univariado, en el cual se asume que los datos vienen de una población normalmente distribuida.

El test es capaz de detectar un outlier a la vez.

Las hipótesis que se testean son las siguientes:

- Ho: No hay outliers en los datos
- Ha: Hay exactamente un outlier en los datos.

El p-valor 1.995367E-05 es significativo con los valores de alpha usuales 0.025, 0.01. Por lo tanto rechazamos la hipótesis nula de que no hay outliers en los datos y aceptamos la hipótesis alternativa de que hay 1 outlier en los datos (el valor 846).

```
valor.de.outlier.Grubbs <- outlier(mydata.numeric)
valor.de.outlier.Grubbs
```

```
## [1] 846
```

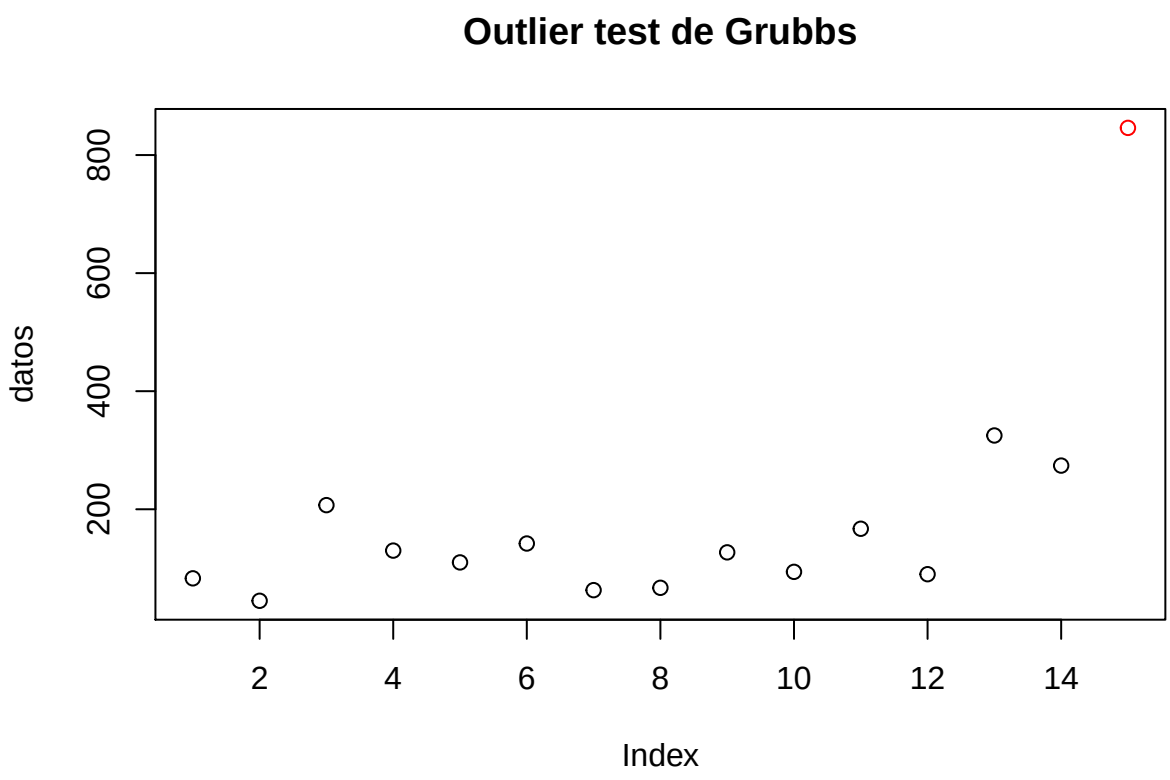
```
indice.de.outlier.Grubbs <- which(mydata.numeric == valor.de.outlier.Grubbs)
indice.de.outlier.Grubbs
```

```
## [1] 15
```

Obtuvimos el valor del outlier con la función *outlier*, luego obtuvimos el índice del outlier chequeando cual índice se corresponde con el valor obtenido.

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

```
##
## N?mero de datos: 15
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



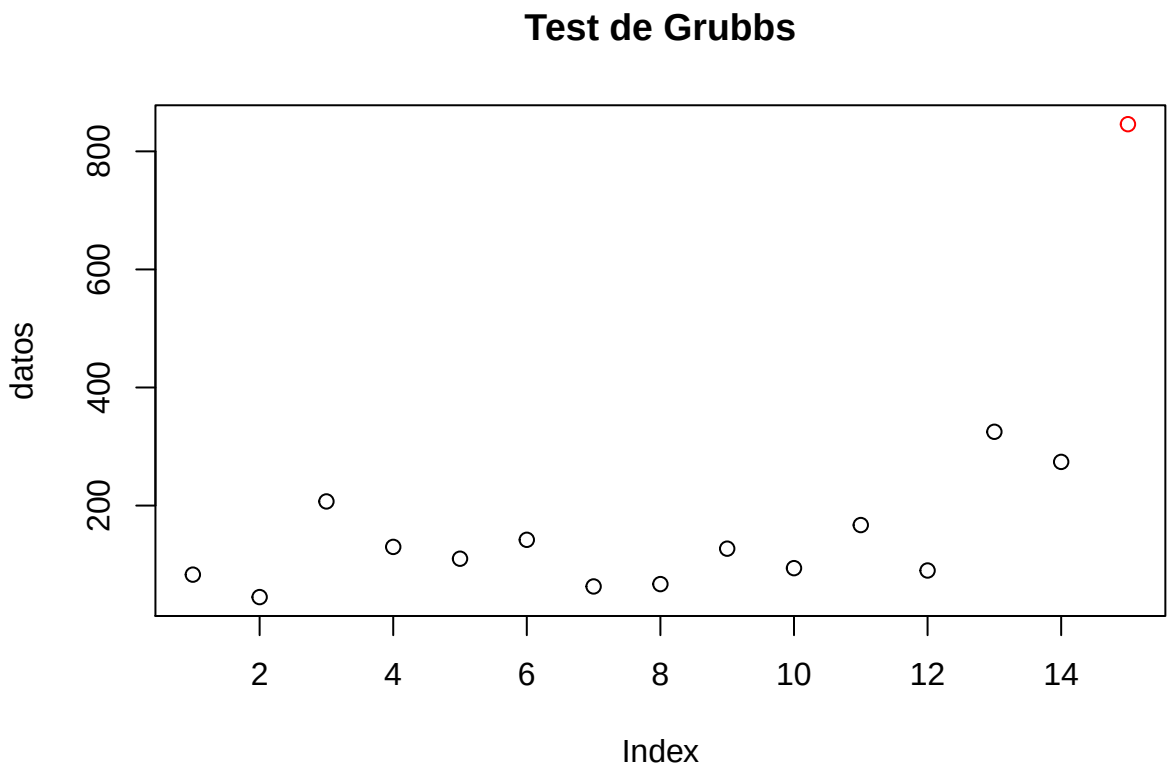
La función MiPlot_Univariate_Outliers grafica los datos y marca el outlier con color rojo. Para esto toma

como parámetros los datos y un vector booleano indicando el valor/es outlier.

El mismo proceso anterior empaquetado en una función

```
MiPlot_resultados_TestGrubbs(mydata.numeric)
```

```
## p.value: 1.995367e-05
## ?ndice de outlier: 15
## Valor del outlier: 846
## N?mero de datos: 15
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



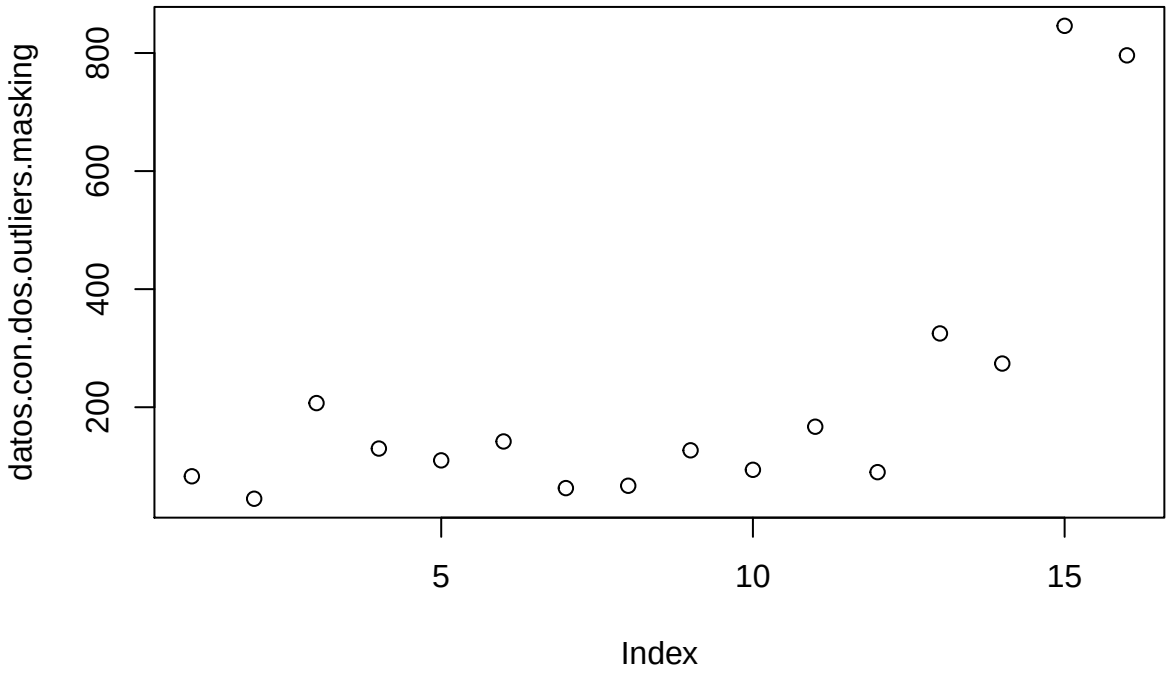
La función MiPlot_resultados_TestGrubbs aplica el test de Grubbs y grafica los resultados. Además muestra por pantalla el p-valor del test, 1.995367E-05 en este caso, el índice del valor outlier detectado, el valor del mismo, y un vector booleano indicando cada indice si es outlier o no.

Genero dos nuevos vectores, uno con dos outliers y otro con varios outliers

```
# Agrego un nuevo outlier
datos.con.dos.outliers.masking <- c(mydata.numeric, (max(mydata.numeric) - 50))
# Datos con varios outliers
datos.con.varios.outliers <- columna
```

Datos.con.dos.outliers.masking

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




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

```
## [1] 0.06436966
```

```
test.de.Grubbs
```

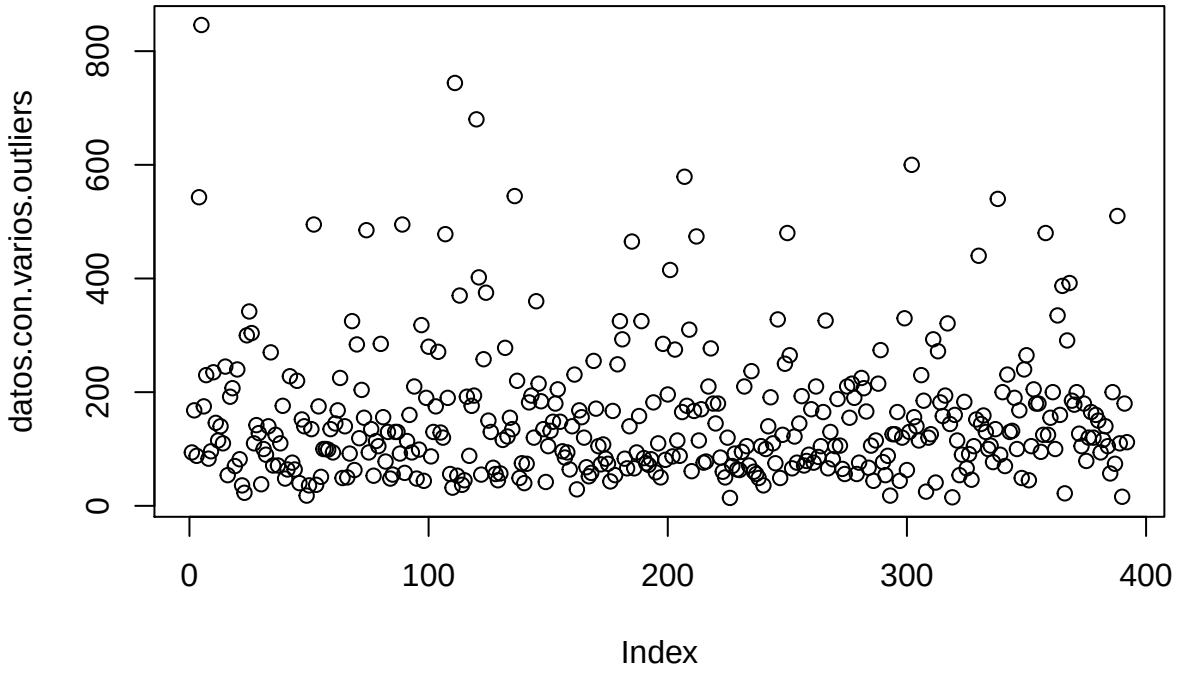
```
##
## Results of Hypothesis Test
## --------------------------
##
## Alternative Hypothesis:          highest value 846 is an outlier
##
## Test Name:                       Grubbs test for one outlier
##
## Data:                            datos.con.dos.outliers.masking
##
## Test Statistics:                 G = 2.5360199
##                                  U = 0.5426562
##
## P-value:                         0.06436966
```

El resultado no es significativo, pvalor: 0.06436966, el test no fue capaz de encontrar un outlier. Sufrimos el efecto de masking. La presencia de dos outliers mueve la media y el desvio estandar hacia ellos, haciendo que

el estadístico no pueda detectarlos como outliers.

Datos con varios outliers

```
plot(datos.con.varios.outliers)
```




```
test.de.Grubbs <- grubbs.test(datos.con.varios.outliers)
test.de.Grubbs$p.value
```

```
## [1] 5.814236e-07
```

```
test.de.Grubbs
```

```
##
## Results of Hypothesis Test
## --------------------------
##
## Alternative Hypothesis:          highest value 846 is an outlier
##
## Test Name:                       Grubbs test for one outlier
##
## Data:                            datos.con.varios.outliers
##
## Test Statistics:                 G = 5.8055711
##                                  U = 0.9135784
##
```

```
## P-value:                          5.814236e-07
```

El test fue capaz de detectar un valor outlier, pvalor: 5.814236E-07, el outlier es el valor 846. Sin embargo en este caso teníamos varios outliers y el test solo nos informa la presencia de uno. Lo que se puede realizar es iterativamente ir eliminando el valor outlier y probando nuevamente hasta que el test no detecte la presencia de outliers.

Test de Rosner

El test de Rosner permite detectar un número de outliers menor o igual a un valor determinado.

En este caso vamos a probar con un valor de 4.

```
test.de.rosner <- rosnerTest(datos.con.dos.outliers.masking, k = 4)
```

```
## Warning in rosnerTest(datos.con.dos.outliers.masking, k = 4): The true Type
## I error may be larger than assumed. See the help file for 'rosnerTest' for
## a table with information on the estimated Type I error level.
```

```
test.de.rosner$all.stats$Outlier
```

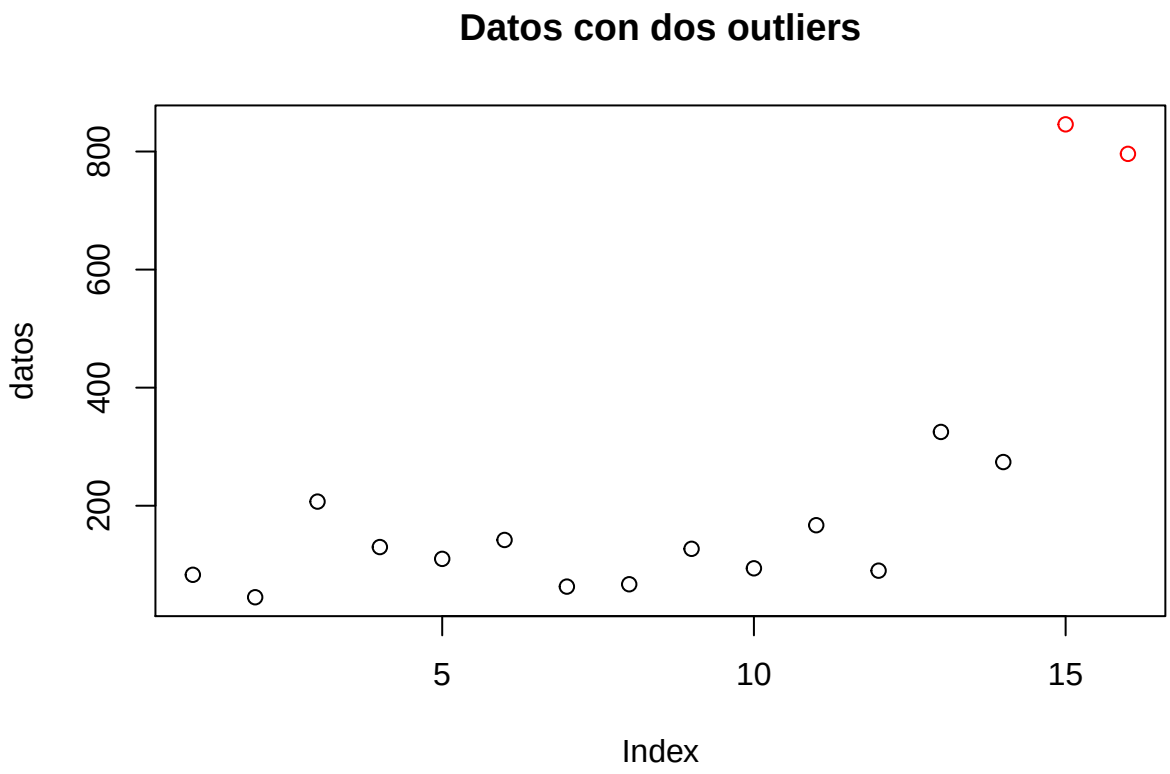
```
## [1]  TRUE  TRUE FALSE FALSE
```

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

```
## [1] 15 16 13 14
```

```
indices.outliers <- test.de.rosner$all.stats$Obs.Num[test.de.rosner$all.stats$Outlier]

MiPlot_Univariate_Outliers(datos.con.dos.outliers.masking,
indices.outliers,
"Datos con dos outliers")
```

```
##
## N?mero de datos: 16
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```

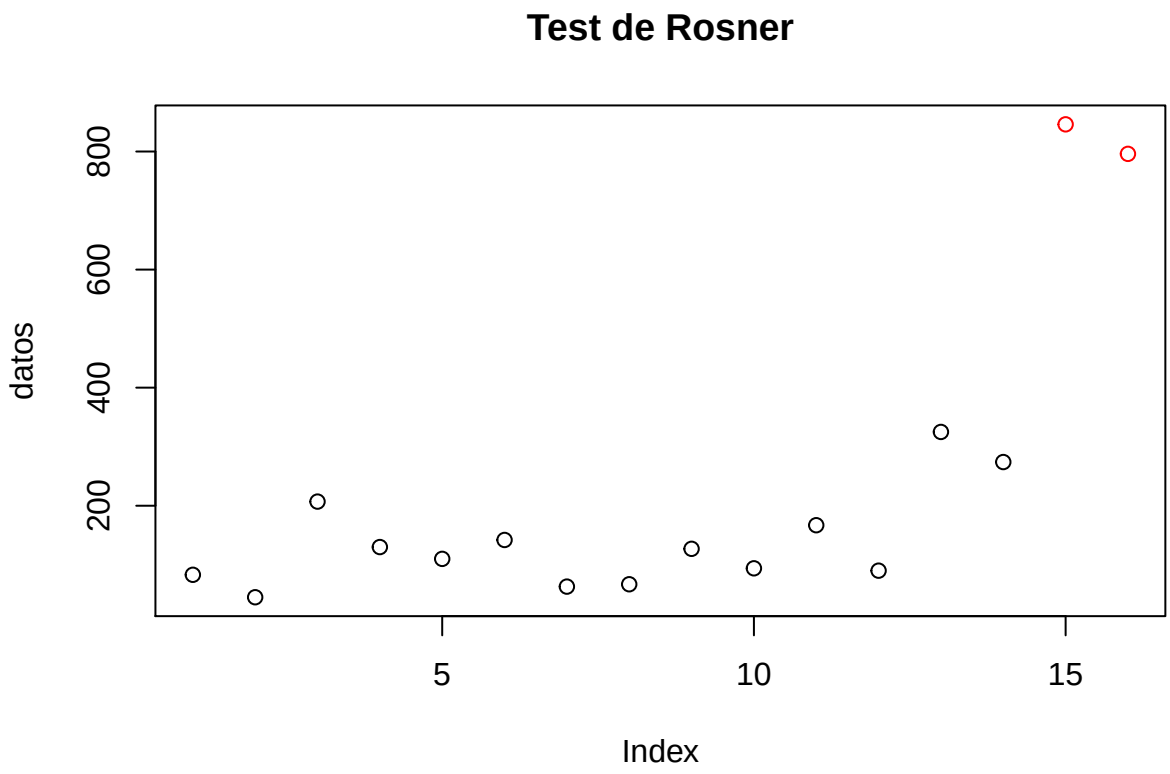


El test es capaz de detectar los dos outliers presentes en los datos. Luego le pasamos los índices de los outliers a la función MiPlot_Univariate_Outliers para graficar los datos y marcar en rojo los outliers.

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

```
## Warning in rosnerTest(datos, k = 4): The true Type I error may be
## larger than assumed. See the help file for 'rosnerTest' for a table with
## information on the estimated Type I error level.
```

```
##
## Test de Rosner
## ?ndices de las k-mayores desviaciones de la media: 15 16 13 14
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE FALSE FALSE
## Los ?ndices de los outliers son: 15 16
## Los valores de los outliers son: 846 796
## N?mero de datos: 16
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



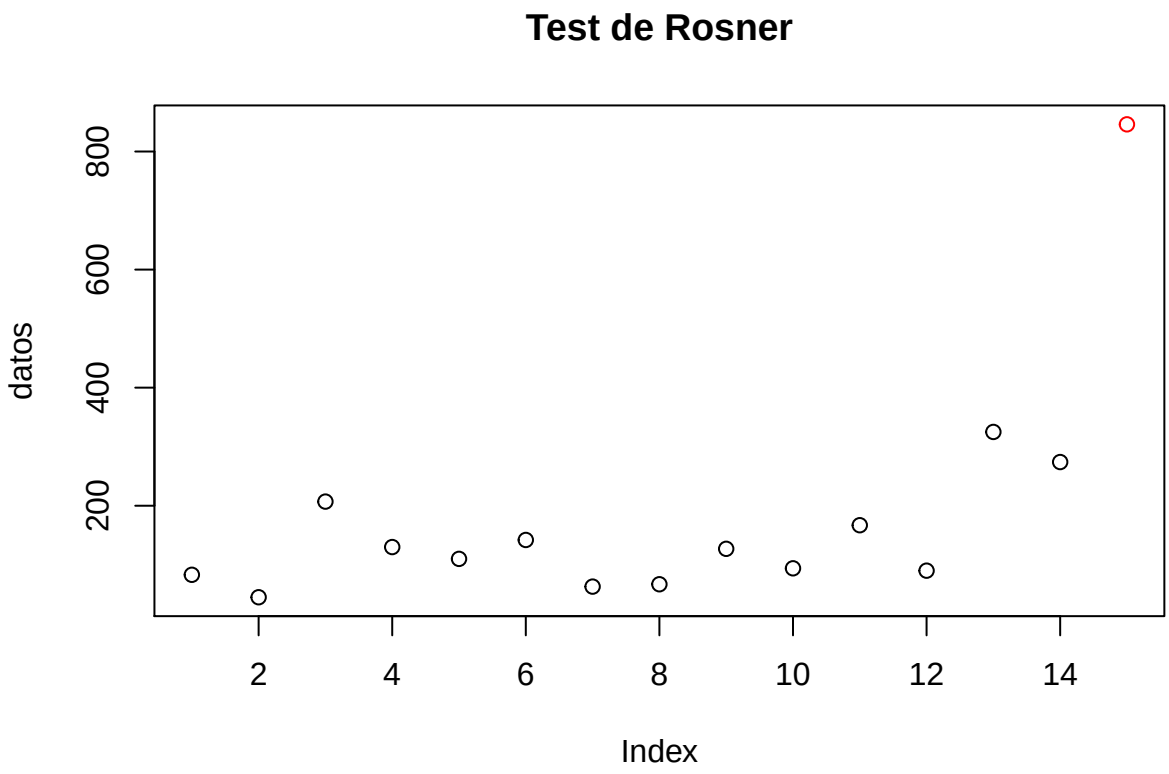
La función MiPlot_resultados_TestRosner automatiza la ejecución del test de Rosner y el gráfico de resultados. Además nos informa la siguiente información reelevante:

- Índices de las observaciones con mayor desviación
- Vector booleano indicando cuales de los índices anterior son outliers.
- Índices de los outliers
- Valores de los outliers
- Número de datos analizados
- Vector booleano indicando para todos los datos analizados cuales son outliers.

```
MiPlot_resultados_TestRosner(datos.con.un.outlier)
```

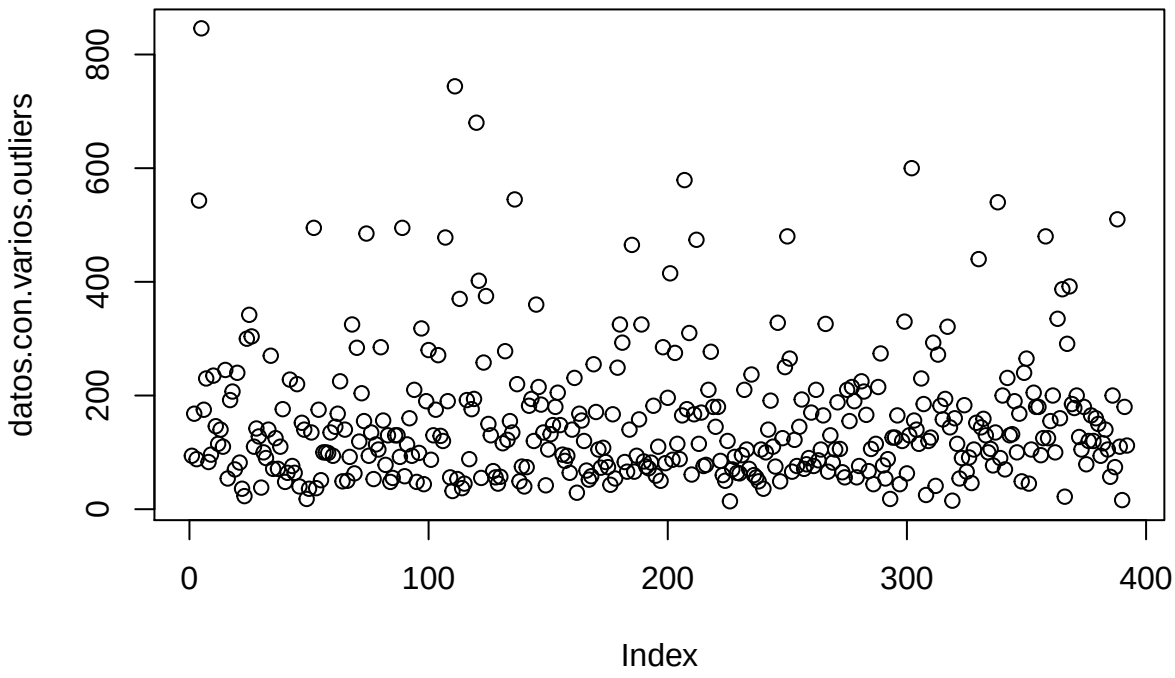
```
## Warning in rosnerTest(datos, k = 4): The true Type I error may be
## larger than assumed. See the help file for 'rosnerTest' for a table with
## information on the estimated Type I error level.
```

```
##
## Test de Rosner
## ?ndices de las k-mayores desviaciones de la media: 15 13 14 3
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE FALSE FALSE FALSE
## Los ?ndices de los outliers son: 15
## Los valores de los outliers son: 846
## N?mero de datos: 15
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



Resultados de la funcion sobre el set de datos que tenía un único outlier, podemos observar como el test de Rosner lo identifica de forma óptima.

```
plot(datos.con.varios.outliers)
```




```
test.de.rosner <- rosnerTest(datos.con.varios.outliers, k = 4)
test.de.rosner$all.stats$Outlier
```

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

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

```
## [1]   5 111 120 302
```

```
# Índices de los outliers
indices.outliers <- test.de.rosner$all.stats$Obs.Num[test.de.rosner$all.stats$Outlier]
indices.outliers
```

```
## [1]   5 111 120 302
```

```
# Valores de los outliers
datos.con.varios.outliers[indices.outliers]
```

```
## [1] 846 744 680 600
```

En este caso al tener varios outliers el dataset, el test de Rosner es capaz de encontrar 4, ya que estamos probando la hipótesis de que el dataset tiene 4 o menos outliers. Los valores de los outliers son 846, 744, 680, 600

Voy a probar con un valor de 30, la idea es ver si puedo encontrar los 25 outliers que fueron encontrados con el método IQR.

```
test.de.rosner <- rosnerTest(datos.con.varios.outliers, k = 30)
```

```
## Warning in rosnerTest(datos.con.varios.outliers, k = 30): The true Type I error may be larger than as
## Although the help file for 'rosnerTest' has a table with information
```

```
## on the estimated Type I error level,
## simulations were not run for k > 10 or k > floor(n/2).
```

```
test.de.rosner$all.stats$Outlier
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

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

```
##  [1]   5 111 120 302 207 136   4 338 388  52  89  74 250 358 107 212 185
## [18] 330 201 121 368 365 124 113 145  25 363 299 246 266
```

```
indices.outliers <- test.de.rosner$all.stats$Obs.Num
```

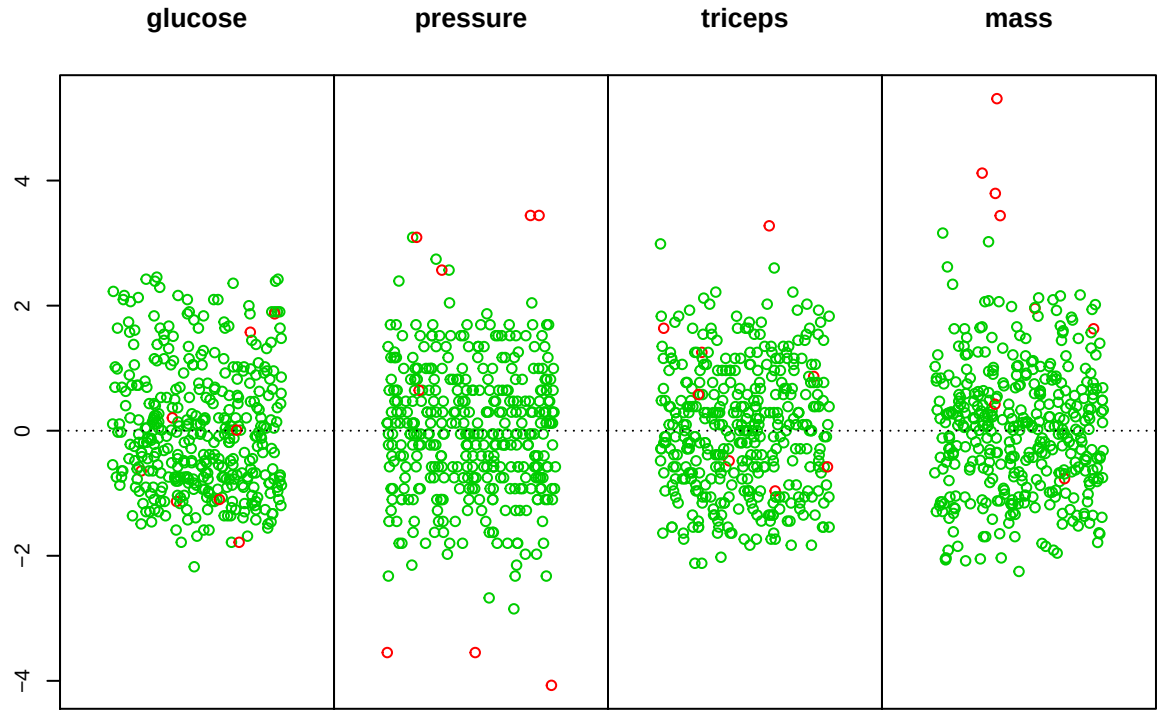
Se encuentran 17 outliers, se recibe una advertencia, que no se calcula el error para valores de k > 10

MULTIVARIATE STATISTICAL OUTLIERS -> Multivariate Normal Distribution –> Mahalanobis

Detección de Outliers multivariantes en distribuciones conjuntas normales aplicando tests basados en la distancia de Mahalanobis

Obtención de los outliers multivariantes

```
# Me quedo con 4 columnas del dataset: glucose, triceps, mass y pressure.
mydata.numeric <- mydata.numeric.backup[,
colnames(mydata.numeric.backup)
%in%
c("glucose", "triceps", "mass", "pressure")]
mydata.numeric.scaled <- mydata.numeric.scaled.backup[,
colnames(mydata.numeric.scaled.backup)
%in%
c("glucose", "triceps", "mass", "pressure")]

set.seed(12)
mvoutlier.plot <- uni.plot(mydata.numeric, symb = FALSE, alpha = 0.05)
```




```
mvoutlier.plot$outliers
```

```
##    S1    S2    S3    S4    S5    S6    S7    S8    S9   S10   S11   S12
## FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##   S13   S14   S15   S16   S17   S18   S19   S20   S21   S22   S23   S24
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##   S25   S26   S27   S28   S29   S30   S31   S32   S33   S34   S35   S36
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   S37   S38   S39   S40   S41   S42   S43   S44   S45   S46   S47   S48
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   S49   S50   S51   S52   S53   S54   S55   S56   S57   S58   S59   S60
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##   S61   S62   S63   S64   S65   S66   S67   S68   S69   S70   S71   S72
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   S73   S74   S75   S76   S77   S78   S79   S80   S81   S82   S83   S84
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   S85   S86   S87   S88   S89   S90   S91   S92   S93   S94   S95   S96
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##   S97   S98   S99  S100  S101  S102  S103  S104  S105  S106  S107  S108
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S109  S110  S111  S112  S113  S114  S115  S116  S117  S118  S119  S120
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  S121  S122  S123  S124  S125  S126  S127  S128  S129  S130  S131  S132
## FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  S133  S134  S135  S136  S137  S138  S139  S140  S141  S142  S143  S144
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
##  S145  S146  S147  S148  S149  S150  S151  S152  S153  S154  S155  S156
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S157  S158  S159  S160  S161  S162  S163  S164  S165  S166  S167  S168
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S169  S170  S171  S172  S173  S174  S175  S176  S177  S178  S179  S180
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S181  S182  S183  S184  S185  S186  S187  S188  S189  S190  S191  S192
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S193  S194  S195  S196  S197  S198  S199  S200  S201  S202  S203  S204
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S205  S206  S207  S208  S209  S210  S211  S212  S213  S214  S215  S216
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S217  S218  S219  S220  S221  S222  S223  S224  S225  S226  S227  S228
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  S229  S230  S231  S232  S233  S234  S235  S236  S237  S238  S239  S240
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S241  S242  S243  S244  S245  S246  S247  S248  S249  S250  S251  S252
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S253  S254  S255  S256  S257  S258  S259  S260  S261  S262  S263  S264
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S265  S266  S267  S268  S269  S270  S271  S272  S273  S274  S275  S276
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S277  S278  S279  S280  S281  S282  S283  S284  S285  S286  S287  S288
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S289  S290  S291  S292  S293  S294  S295  S296  S297  S298  S299  S300
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S301  S302  S303  S304  S305  S306  S307  S308  S309  S310  S311  S312
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  S313  S314  S315  S316  S317  S318  S319  S320  S321  S322  S323  S324
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S325  S326  S327  S328  S329  S330  S331  S332  S333  S334  S335  S336
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S337  S338  S339  S340  S341  S342  S343  S344  S345  S346  S347  S348
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
##  S349  S350  S351  S352  S353  S354  S355  S356  S357  S358  S359  S360
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S361  S362  S363  S364  S365  S366  S367  S368  S369  S370  S371  S372
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S373  S374  S375  S376  S377  S378  S379  S380  S381  S382  S383  S384
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  S385  S386  S387  S388  S389  S390  S391  S392
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

La función uni.plot obtiene 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 (minimum covariance determinant).

No hay que normalizar los datos ya que la distancia de Mahalanobis está diseñada, precisamente para evitar el problema de la escala.

El grafico nos muestra los outliers univariantes para cada columna del dataset, pero sin su etiqueta.

```
is.MCD.outlier <- mvoutlier.plot$outliers
numero.de.outliers.MCD <- length(mvoutlier.plot$outliers[mvoutlier.plot$outliers == TRUE])
```

```
indices.de.outliers.multivariantes.MCD <- which(mvoutlier.plot$outliers == TRUE)
```

El dataset tiene 14 outliers multivariantes segun el calculo de la distancia de Mahalanobis.

Las siguientes observaciones son outliers multivariantes:

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

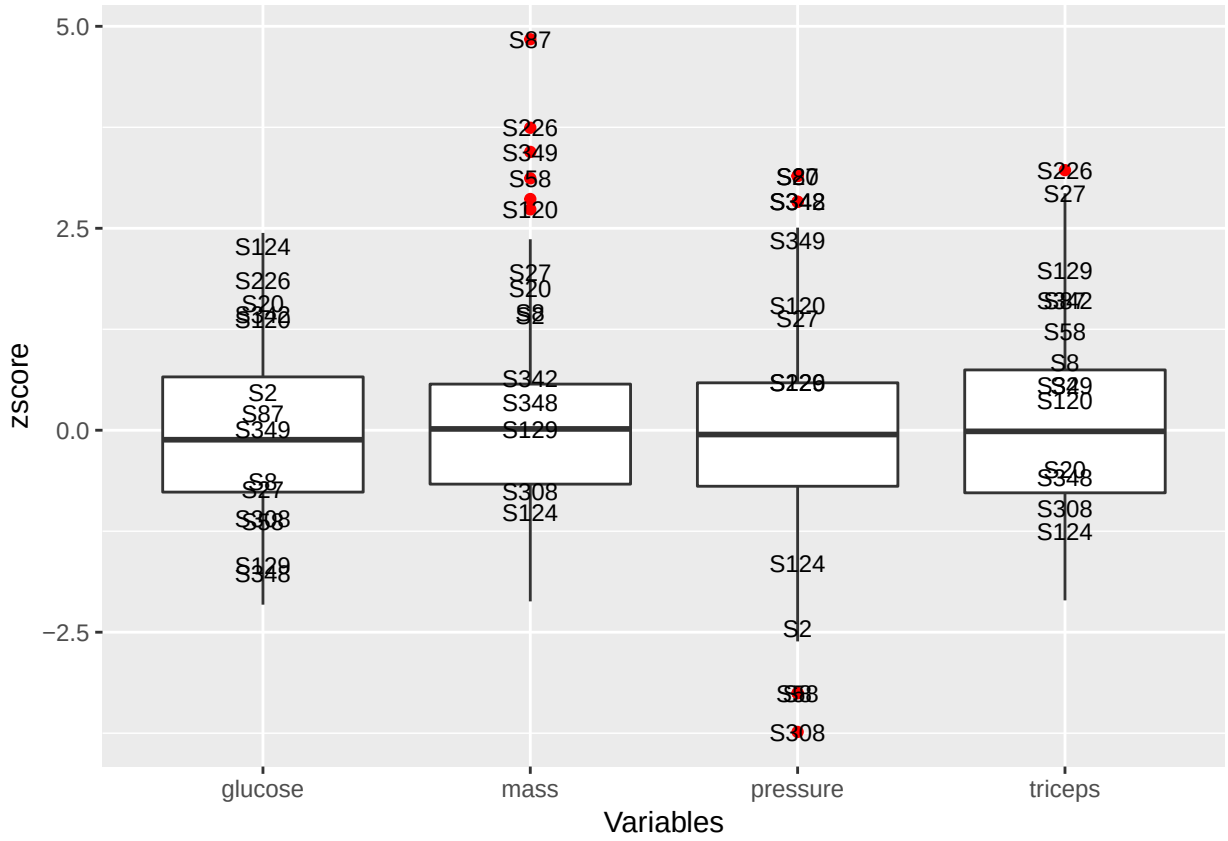
```
##   S2   S8  S20  S27  S58  S87 S120 S124 S129 S226 S308 S342 S348 S349
##    2    8   20   27   58   87  120  124  129  226  308  342  348  349
```

Obtenemos los valores normalizados para los outliers multivariantes y mostramos un boxplots con las muestras etiquetadas.

```
data.frame.solo.outliers <- mydata.numeric.scaled[is.MCD.outlier, ]
data.frame.solo.outliers
```

```
##          glucose   pressure    triceps        mass
## S2    0.46571891 -2.4538285  0.5567094  1.42490909
## S8   -0.63600306 -3.2540787  0.8419775  1.45336807
## S20   1.56744087  3.1479231 -0.4892736  1.75218734
## S27  -0.73321383  1.3873726  2.9339434  1.95140019
## S58  -1.12205687 -3.2540787  1.2223349  3.11821830
## S87   0.20649021  3.1479231  1.6026923  4.83998648
## S120  1.37301935  1.5474226  0.3665307  2.73402209
## S124  2.28031980 -1.6535783 -1.2499884 -1.02256303
## S129 -1.67291786  0.5871224  1.9830498  0.01618967
## S226  1.85907316  0.5871224  3.2192114  3.74431582
## S308 -1.08965329 -3.7342288 -0.9647204 -0.75220274
## S342  1.43782653  2.8278230  1.6026923  0.64228719
## S348 -1.77012862  2.8278230 -0.5843629  0.34346792
## S349  0.01206868  2.3476728  0.5567094  3.44549655
```

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

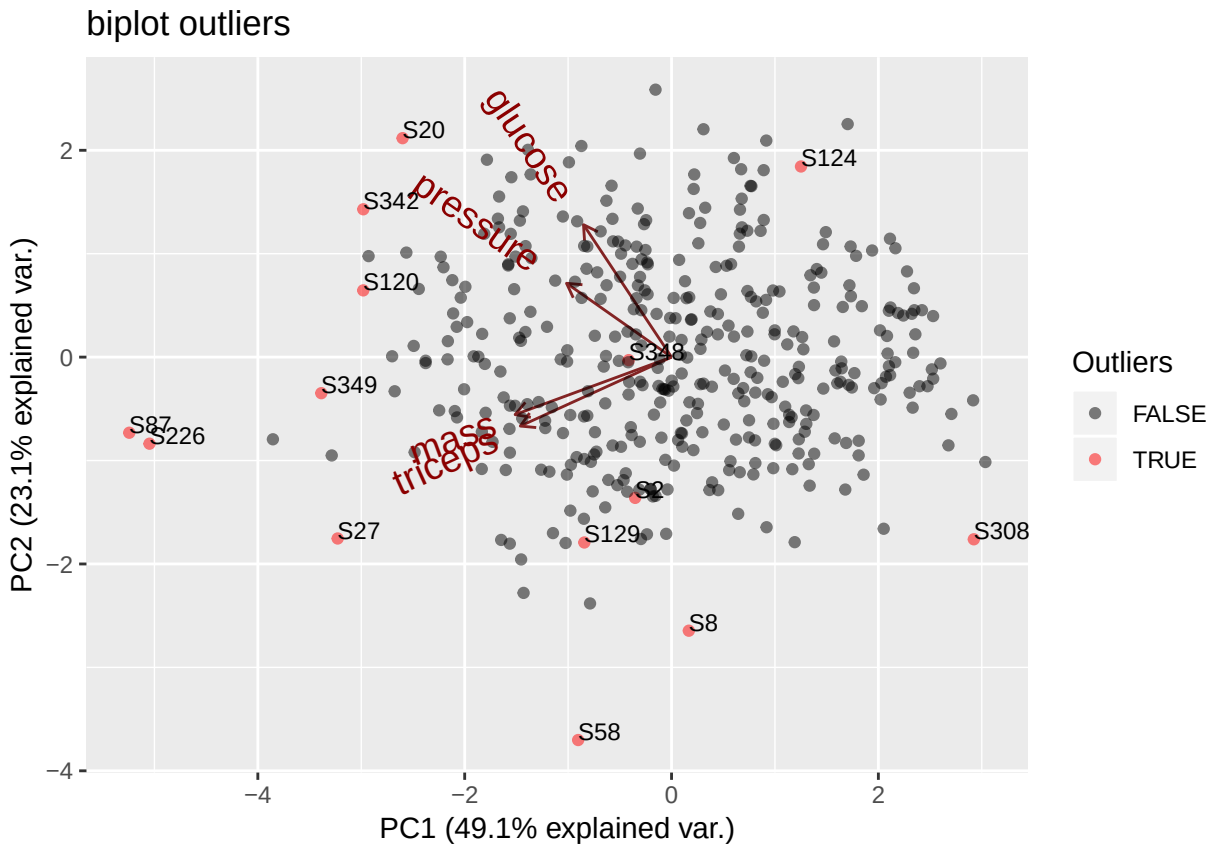


A partir de la inspección del dataframe que nos indica que muestras son outliers multivariantes, podemos aprecir cual es el valor que más contribuye a ser un outlier en cada muestra, considerando que observamos el desvío del valor con respecto a la media. Aún asi, hay casos en que es complicado distinguir la relación entre variables que genera el outlier multivariante.

Al observar los boxplots podemos discernir que muchas muestras son outliers multivariantes porque tambien son univariantes, pero hay muestras como ser S124 que no es un outlier univariante para ninguna variable, sin embargo es un outlier multivariante. Siendo en la variable *glucose* donde es mas extremo.

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

```
##  S2      S8            S20       S27                          S58                          S8
```



Observando el biplot podemos llegar a la misma conclusión con respecto a la muestra S124. No es un outlier univariante para ninguna columna, sino que es la combinación de valores para diferentes variables lo que lo convierte en outlier.

El biplot explica casi el 75% de la variabilidad de los datos.

```
indices.de.outliers.en.alguna.columna <- vector_claves_outliers_IQR_en_alguna_columna(mydata.numeric)

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

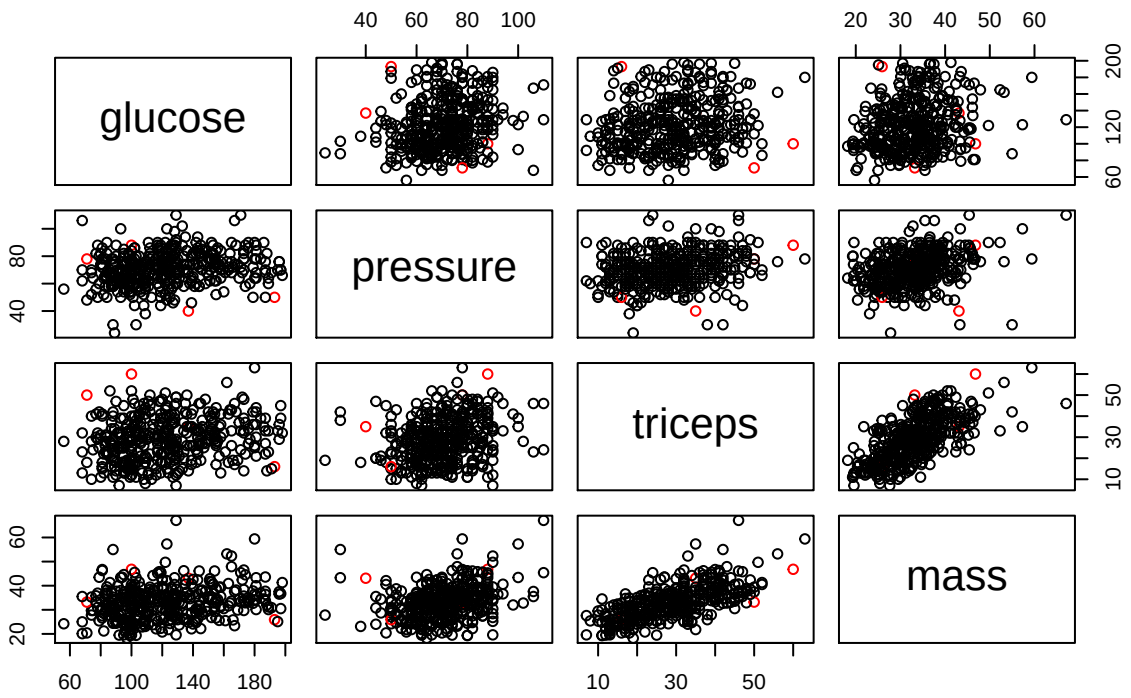
```
## [1]   2  27 124 129
```

Podemos observar que los índices 2, 27, 124, 129 se corresponden a dichas muestras son outliers multivariantes, pero no univariantes en ninguna de las 4 columnas analizadas. Es decir que si miramos cada columna por separado, estos cuatro valores no seran outliers en ninguna de ellas, pero la combinacion de valoers en diferentes columnas los convierte en outliers.

Vamos a graficar estos 4 outliers.

```
MiPlot_Univariate_Outliers(mydata.numeric,
indices.de.outliers.multivariantes.MCD.pero.no.1variantes,
"Outliers multivariantes pero no 1 variantes")
```

```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
```

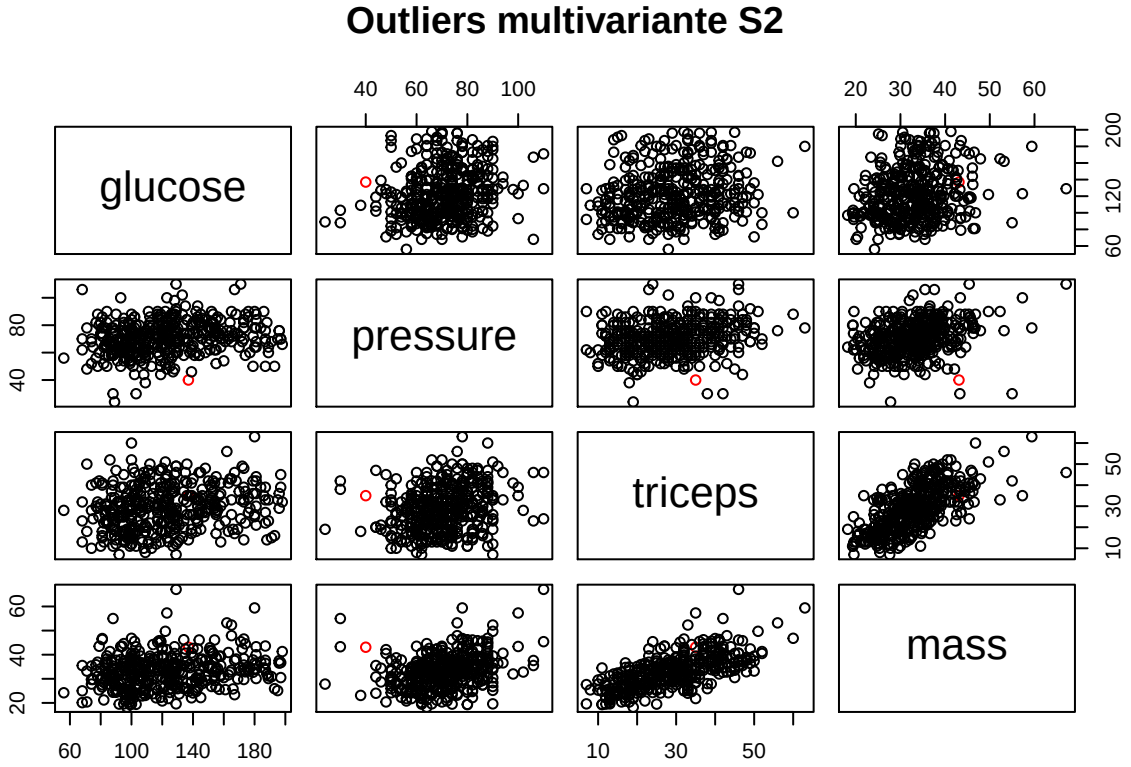


Vamos a realizarlo para cada uno por separado:

- S2

```
MiPlot_Univariate_Outliers(mydata.numeric,
2,
"Outliers multivariante S2")
```

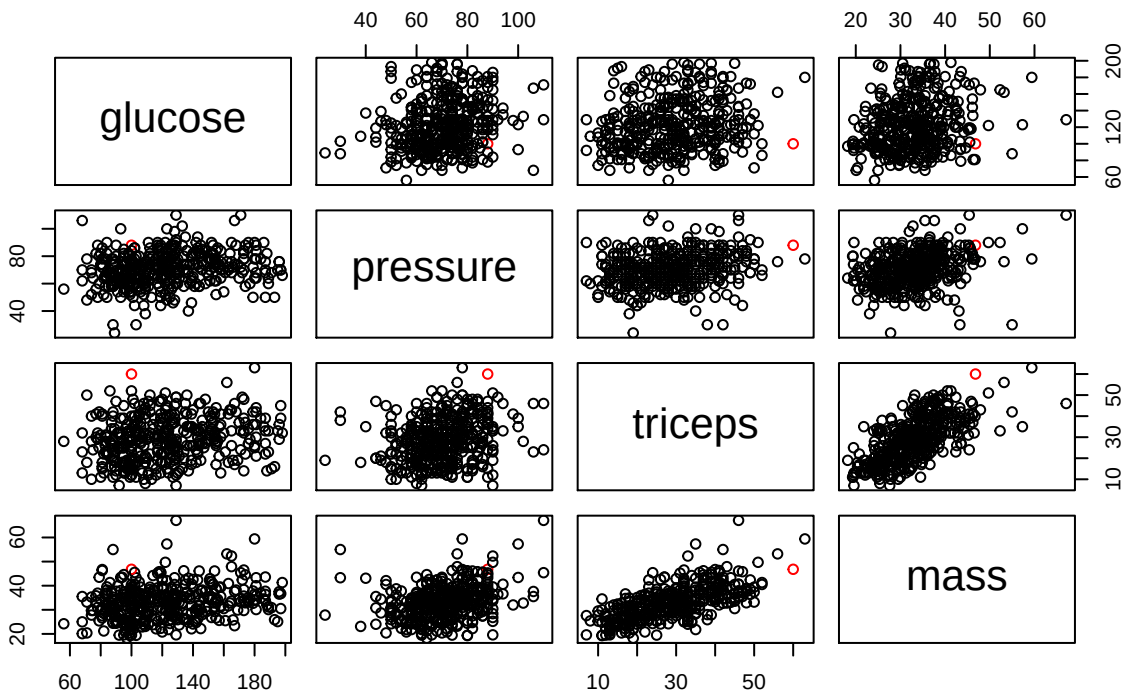
```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
```



- S27

```
MiPlot_Univariate_Outliers(mydata.numeric,
27,
"Outliers multivariante S27")
```

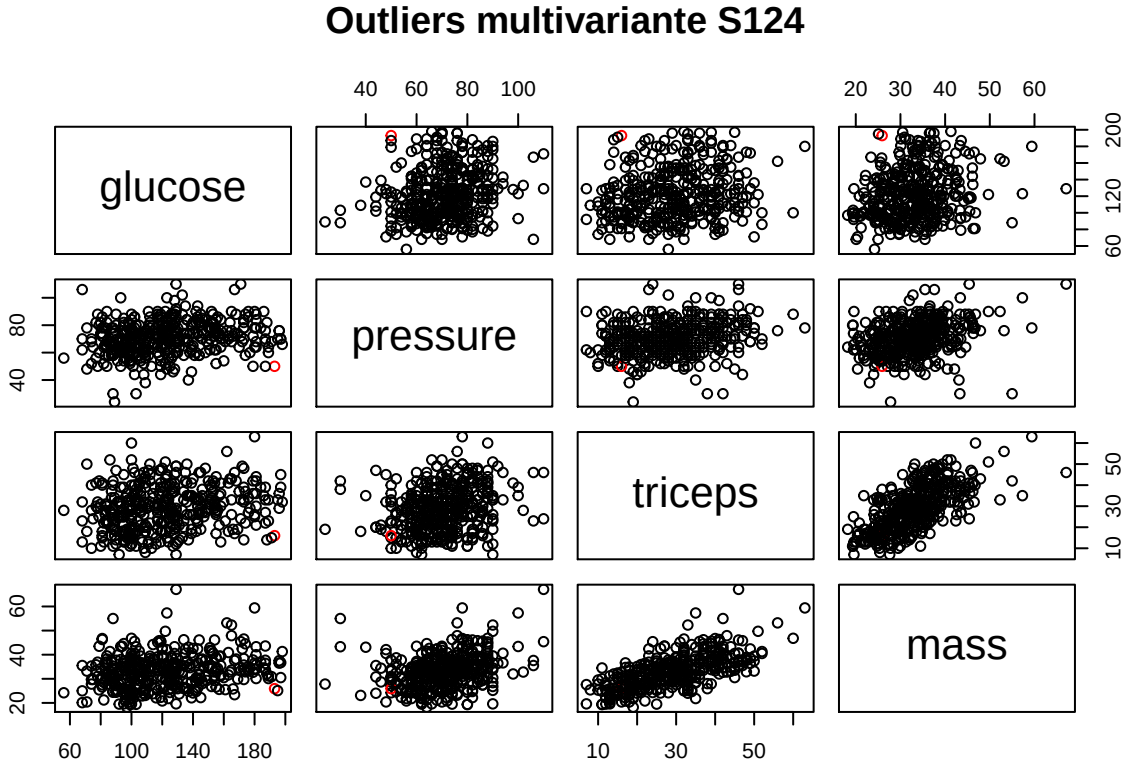
```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FAL
```



- S124

```
MiPlot_Univariate_Outliers(mydata.numeric,
124,
"Outliers multivariante S124")
```

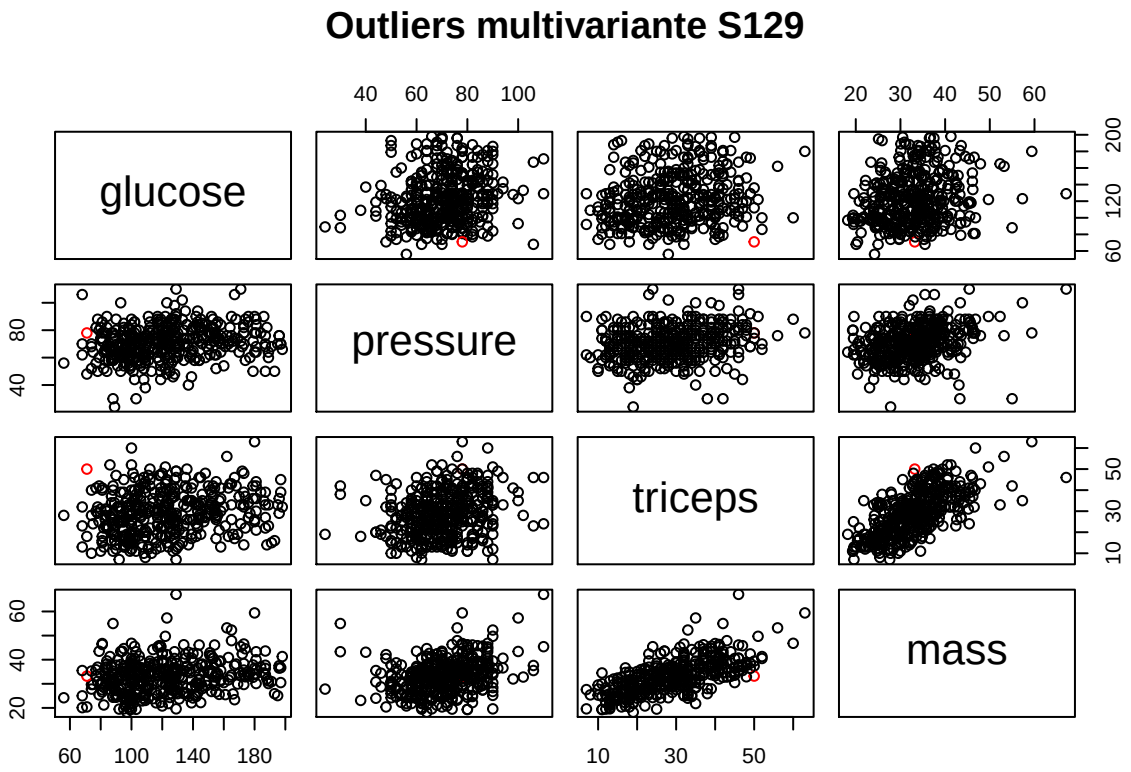
```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



- S129

```
MiPlot_Univariate_Outliers(mydata.numeric,
129,
"Outliers multivariante S129")
```

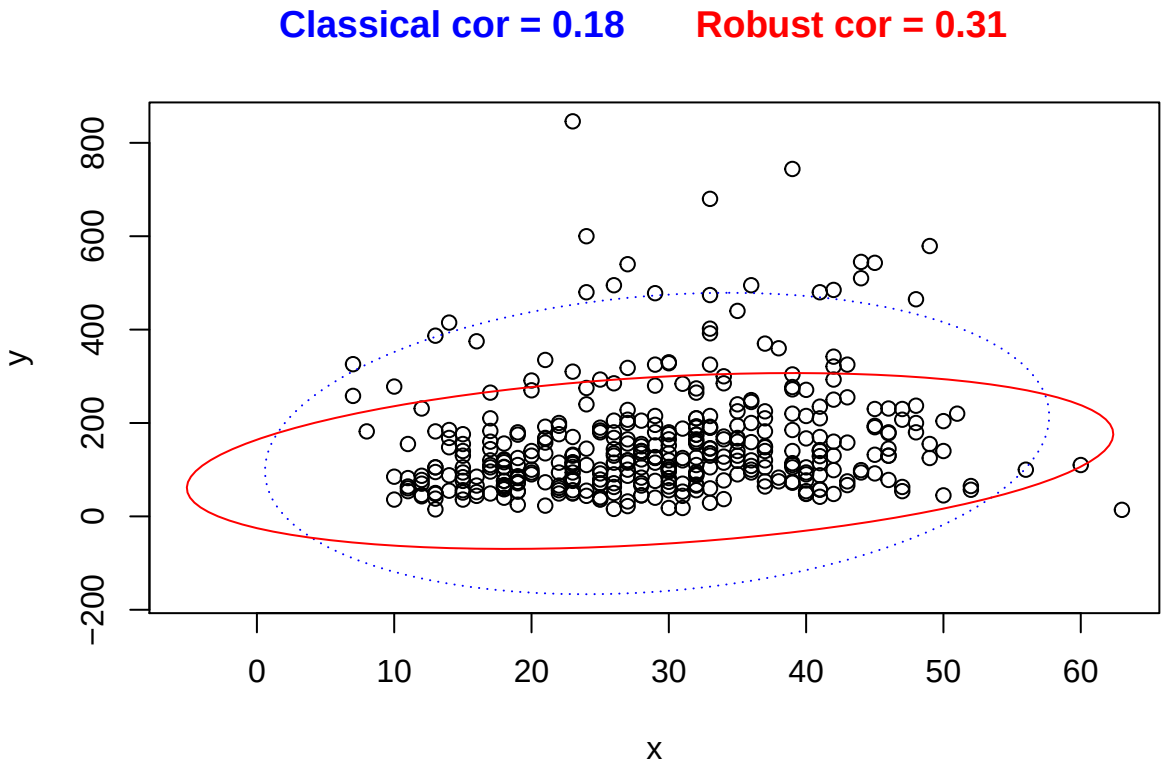
```
##
## N?mero de datos: 392
## ?Qui?n es outlier?: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```



MULTIVARIATE STATISTICAL OUTLIERS -> LOF

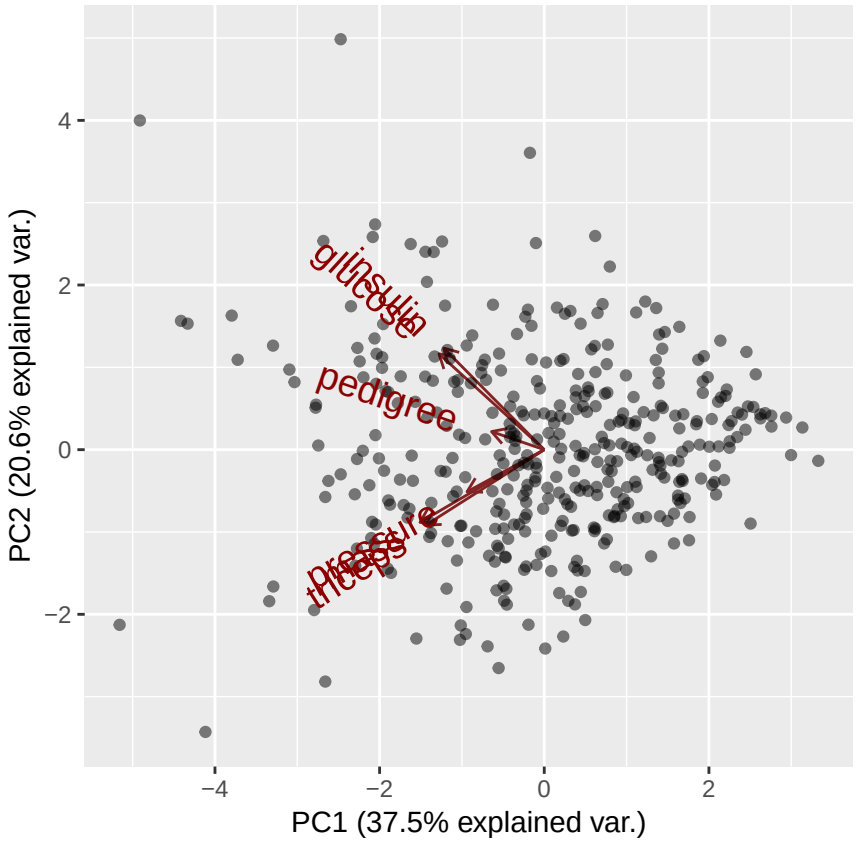
Detección de Outliers multivariantes según el método LOF.

```
mis.datos.numericos <- mydata.numeric.backup
mis.datos.numericos.normalizados <- mydata.numeric.scaled.backup

corr.plot(mis.datos.numericos[,3], mis.datos.numericos[,4])
```



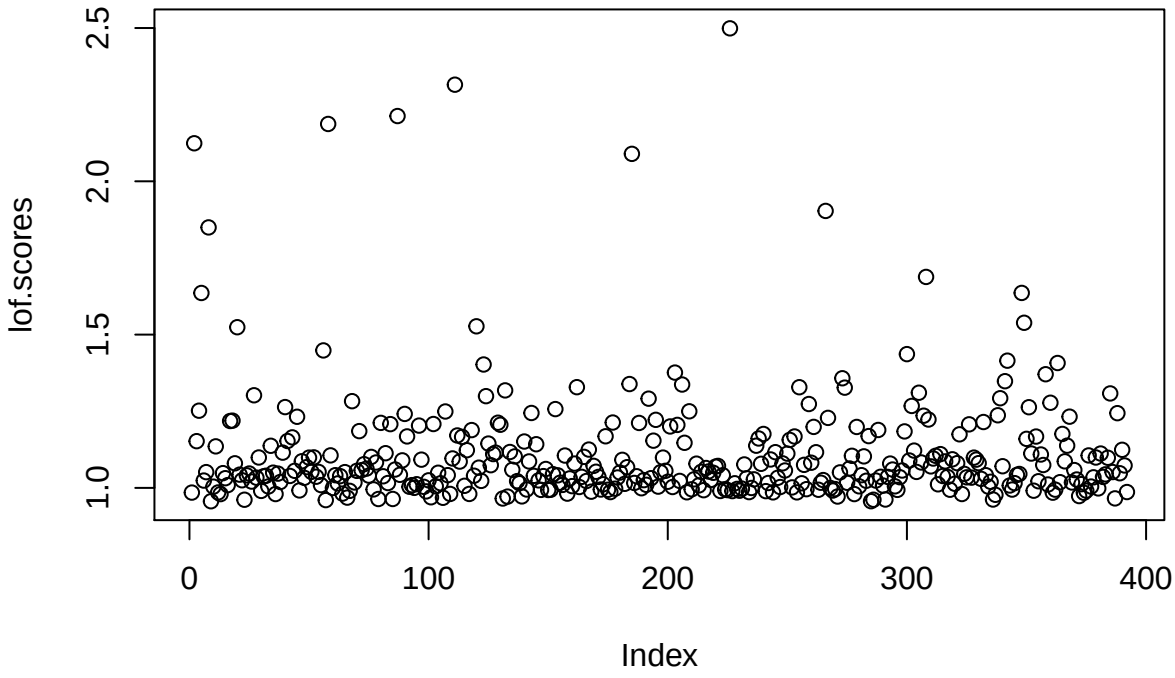

```
## $cor.cla
## [1] 0.1821991
##
## $cor.rob
## [1] 0.307282
```

```
MiBiplot(mis.datos.numericos)
```



DISTANCE BASED OUTLIERS (LOF)

```
numero.de.vecinos.lof = 10
lof.scores <- lofactor(mis.datos.numericos.normalizados, numero.de.vecinos.lof)
plot(lof.scores)
```



Establecí en 10 el número de vecinos a considerar para el calculo de los scores de LOF.

La función lofactor devuelve un vector con los scores de LOF de todos los registros

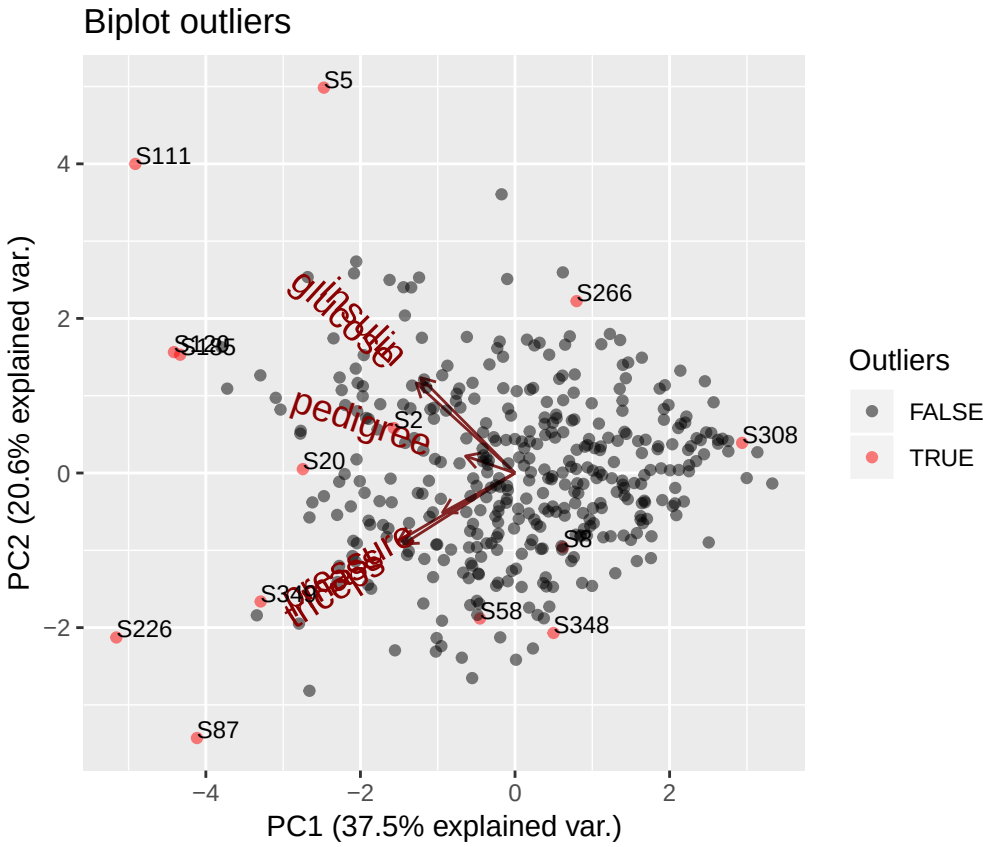
Asumimos que los valores con un score LOF >= 1.5 son outliers.

```
numero.de.outliers = sum(lof.scores >= 1.5)
```

Hay 14 outliers de acuerdo al score LOF, considerando outliers aquellos con un score LOF >= 1.5.

```
indices.de.lof.outliers.ordenados <- sort(lof.scores, index.return = TRUE,
                                          decreasing = TRUE)$ix
indices.de.lof.top.outliers <- indices.de.lof.outliers.ordenados[1:numero.de.outliers]
indices.de.lof.top.outliers.reformat <- paste("S",
                                              indices.de.lof.top.outliers, sep = "")
is.lof.outlier <- row.names(mis.datos.numericos) %in%
  indices.de.lof.top.outliers.reformat
MiBiPlot_Multivariate_Outliers(mis.datos.numericos.normalizados,
                               is.lof.outlier, "Biplot outliers")
```

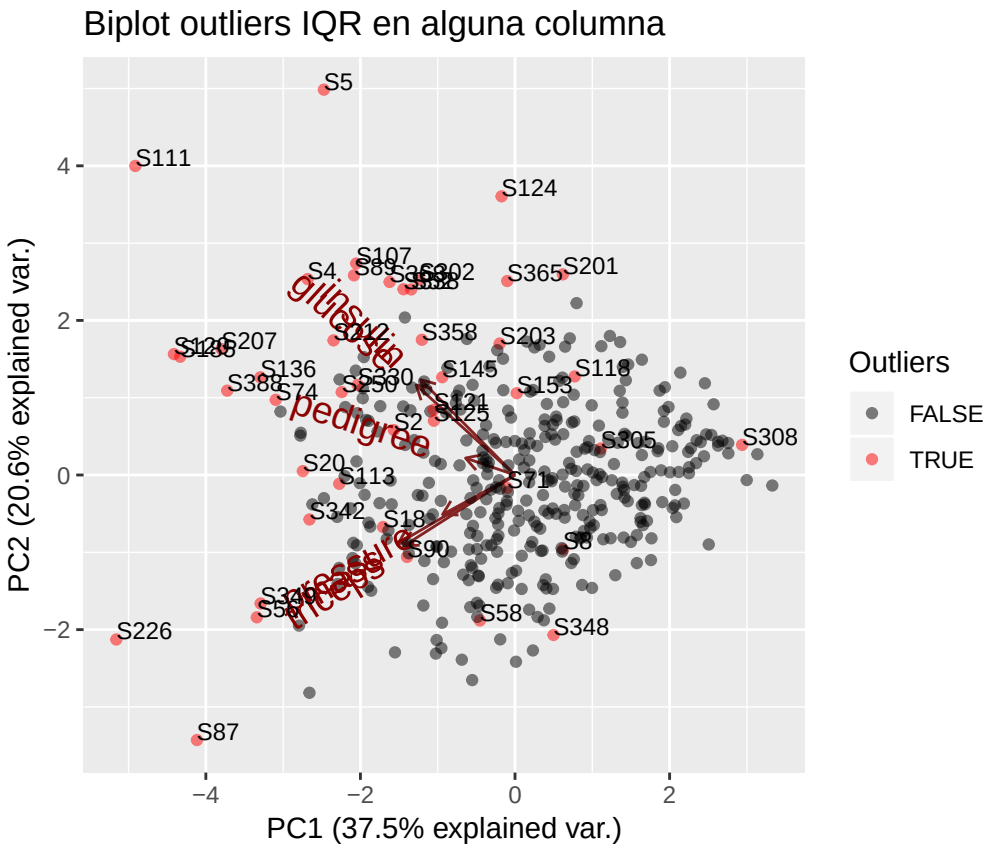
```
##  S2   S5   S8           S20                                  S58                                  S8
```



Obtuve el indice de los outliers y luego grafique un biplot. Hay algunos valores como S8, S348, S58, S266 que no parecen ser outliers univariantes. Lo podemos chequear.

```
vector.claves.outliers.IQR.en.alguna.columna <-
  vector_claves_outliers_IQR_en_alguna_columna(mis.datos.numericos)
vector.es.outlier.IQR.en.alguna.columna <-
  vector_es_outlier_IQR_en_alguna_columna(mis.datos.numericos)
MiBiPlot_Multivariate_Outliers(mis.datos.numericos.normalizados,
                               vector.es.outlier.IQR.en.alguna.columna,
                               "Biplot outliers IQR en alguna columna")
```

```
##  S2  S4 S5   S8          S18  S20                                S52    S56  S58           S71   S
```




```
indices.de.outliers.multivariantes.LOF.pero.no.1variantes <-
  setdiff(indices.de.lof.top.outliers,
vector.claves.outliers.IQR.en.alguna.columna)
indices.de.outliers.multivariantes.LOF.pero.no.1variantes
```

```
## [1] 266
```

266 es el valor que no es un outlier univariante y si es un outlier LOF. Se corresponde a la muestra S266.

MULTIVARIATE STATISTICAL OUTLIERS. CLUSTERING OUTLIERS

Detección de Outliers multivariantes según los métodos basados en clustering.

```
numero.de.outliers   = 5
numero.de.clusters   = 2

set.seed(2)  # Para establecer la semilla para la primera iteraci?n de kmeans
```

kmeans

```
modelo.kmeans <- kmeans(mis.datos.numericos.normalizados, centers = numero.de.clusters)
indices.clustering.pima <- modelo.kmeans$cluster
```

```
centroides.normalizados.pima <- modelo.kmeans$centers
indices.clustering.pima
```

```
##   S1   S2   S3   S4   S5   S6   S7   S8   S9  S10  S11  S12  S13  S14  S15
##    2    1    2    1    1    2    1    2    2    1    1    2    2    2    1
##  S16  S17  S18  S19  S20  S21  S22  S23  S24  S25  S26  S27  S28  S29  S30
##    2    2    1    2    1    2    2    2    1    1    1    1    2    2    2
##  S31  S32  S33  S34  S35  S36  S37  S38  S39  S40  S41  S42  S43  S44  S45
##    2    2    2    1    2    2    2    2    2    2    2    2    2    2    1
##  S46  S47  S48  S49  S50  S51  S52  S53  S54  S55  S56  S57  S58  S59  S60
##    2    2    2    2    2    1    1    2    2    2    1    2    2    2    2
##  S61  S62  S63  S64  S65  S66  S67  S68  S69  S70  S71  S72  S73  S74  S75
##    2    2    1    2    2    2    2    2    2    1    2    1    1    1    2
##  S76  S77  S78  S79  S80  S81  S82  S83  S84  S85  S86  S87  S88  S89  S90
##    2    2    1    2    1    2    2    2    2    2    1    1    2    1    1
##  S91  S92  S93  S94  S95  S96  S97  S98  S99 S100 S101 S102 S103 S104 S105
##    2    1    2    1    2    2    2    2    2    1    2    2    2    1    2
## S106 S107 S108 S109 S110 S111 S112 S113 S114 S115 S116 S117 S118 S119 S120
##    2    1    2    2    2    1    2    1    2    2    1    2    2    1    1
## S121 S122 S123 S124 S125 S126 S127 S128 S129 S130 S131 S132 S133 S134 S135
##    1    2    2    1    1    2    2    2    2    1    2    2    2    2    2
## S136 S137 S138 S139 S140 S141 S142 S143 S144 S145 S146 S147 S148 S149 S150
##    1    1    2    2    2    2    1    1    2    1    1    2    2    2    2
## S151 S152 S153 S154 S155 S156 S157 S158 S159 S160 S161 S162 S163 S164 S165
##    2    2    2    2    2    2    2    2    2    2    2    2    2    2    1
## S166 S167 S168 S169 S170 S171 S172 S173 S174 S175 S176 S177 S178 S179 S180
##    2    2    2    1    1    2    2    1    2    2    2    2    2    1    1
## S181 S182 S183 S184 S185 S186 S187 S188 S189 S190 S191 S192 S193 S194 S195
##    1    2    2    1    1    2    2    2    1    2    2    1    2    2    2
## S196 S197 S198 S199 S200 S201 S202 S203 S204 S205 S206 S207 S208 S209 S210
##    2    2    1    2    2    2    2    2    2    1    2    1    2    1    2
## S211 S212 S213 S214 S215 S216 S217 S218 S219 S220 S221 S222 S223 S224 S225
##    2    1    2    1    2    2    1    1    1    1    2    2    2    2    2
## S226 S227 S228 S229 S230 S231 S232 S233 S234 S235 S236 S237 S238 S239 S240
##    1    2    2    2    2    2    2    2    2    1    2    2    2    2    2
## S241 S242 S243 S244 S245 S246 S247 S248 S249 S250 S251 S252 S253 S254 S255
##    2    1    1    2    2    1    2    2    1    1    1    2    2    2    1
## S256 S257 S258 S259 S260 S261 S262 S263 S264 S265 S266 S267 S268 S269 S270
##    1    2    2    1    2    2    2    2    2    1    2    2    2    2    2
## S271 S272 S273 S274 S275 S276 S277 S278 S279 S280 S281 S282 S283 S284 S285
##    2    2    1    2    1    1    1    2    2    2    1    1    2    1    2
## S286 S287 S288 S289 S290 S291 S292 S293 S294 S295 S296 S297 S298 S299 S300
##    2    2    2    1    2    2    2    2    2    2    2    2    2    1    2
## S301 S302 S303 S304 S305 S306 S307 S308 S309 S310 S311 S312 S313 S314 S315
##    2    1    1    1    2    1    1    2    2    1    1    2    1    2    2
## S316 S317 S318 S319 S320 S321 S322 S323 S324 S325 S326 S327 S328 S329 S330
##    2    1    2    2    1    2    2    2    2    2    2    2    2    2    1
## S331 S332 S333 S334 S335 S336 S337 S338 S339 S340 S341 S342 S343 S344 S345
##    2    1    2    2    2    2    2    1    2    1    2    1    1    1    2
## S346 S347 S348 S349 S350 S351 S352 S353 S354 S355 S356 S357 S358 S359 S360
##    2    1    2    1    2    2    2    2    2    1    2    1    1    2    2
## S361 S362 S363 S364 S365 S366 S367 S368 S369 S370 S371 S372 S373 S374 S375
##    2    2    2    2    2    2    2    1    1    2    2    2    2    2    2
## S376 S377 S378 S379 S380 S381 S382 S383 S384 S385 S386 S387 S388 S389 S390
```

```
##    1    2    2    2    1    2    2    1    2    1    1    2    1    1    2
## S391 S392
##    2    2
```

```
centroides.normalizados.pima
```

```
##      glucose   pressure    triceps    insulin       mass   pedigree
## 1  0.8951992  0.5963056  0.7663736  0.8138774  0.7480753  0.3807965
## 2 -0.4044974 -0.2694418 -0.3462873 -0.3677520 -0.3380192 -0.1720636
```

Se realizo un clustering con k-means con un parámetro de k = 2. Se obtuvieron las asignaciones de a que grupo pertenece cada valor en la variable *indices.clustering.pima*, y los centroides de cada grupo en la variable *centroides.normalizados.pima*.

```
# Función para calcular distancia euclídea a los centroides
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.pima <- distancias_a_centroides(mis.datos.numericos.normalizados,
indices.clustering.pima,
centroides.normalizados.pima)

top.outliers.pima <- order(dist.centroides.pima, decreasing = TRUE)[1:numero.de.outliers]
top.outliers.pima
```

```
## [1] 226 111   2   5  87
```

Se calculo la distancia de cada valor, al centroide del grupo al que fue asignado en el proceso de clustering. La distancia calculada es la **distancia euclídea**. Luego se obtuvieron los top outliers, previamente se definio que se considerarían como outliers los 5 valores más alejados del centroide de su cluster. Los índices de los outliers son los siguientes 226, 111, 2, 5, 87.

```
# Funcion que automatiza la búsqueda de outliers
top_clustering_outliers <- function(datos.normalizados,
indices.asignacion.clustering,
datos.centroides.normalizados,
numero.de.outliers) {

dist.centroides <- distancias_a_centroides(datos.normalizados,
indices.asignacion.clustering,
datos.centroides.normalizados)

top.outliers <- order(dist.centroides, decreasing = TRUE)[1:numero.de.outliers]
res <- list(indices = top.outliers, distancias = dist.centroides[top.outliers])
}
```

```
top.outliers.kmeans <- top_clustering_outliers(mis.datos.numericos.normalizados,
indices.clustering.pima,
centroides.normalizados.pima,
numero.de.outliers)

top.outliers.kmeans$indices
```

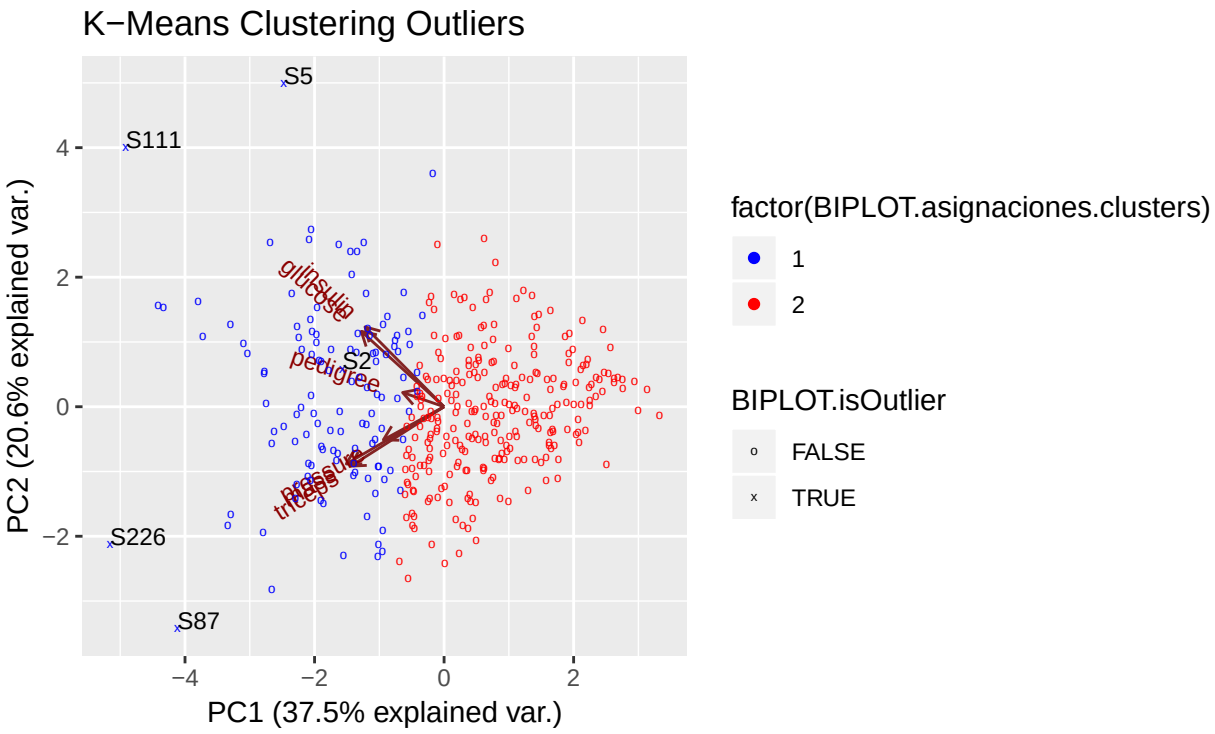
```
## [1] 226 111   2   5  87
```

```
top.outliers.kmeans$distancias
```

```
##     S226     S111       S2       S5      S87
## 6.787530 6.585831 5.731553 5.687426 5.141907
```

La función top_clustering_outliers automatiza la búsqueda de outliers de acuerdo al método de clustering. Esta función devuelve una lista con los índices y las distancias a sus centroides de los outliers. Esta información la imprimimos por pantalla.

```
numero.de.datos   = nrow(mis.datos.numericos)
is.kmeans.outlier = rep(FALSE, numero.de.datos)
is.kmeans.outlier[top.outliers.kmeans$indices] = TRUE
# is.kmeans.outlier[top.outliers.kmeans.distancia.relativa] = TRUE


BIPLOT.isOutlier             = is.kmeans.outlier
# BIPLOT.cluster.colors        = c("blue","red","brown")     # Tantos colores como diga numero.de.clust
BIPLOT.cluster.colors        = c("blue","red")     # Tantos colores como diga numero.de.clusters
BIPLOT.asignaciones.clusters = indices.clustering.pima
MiBiPlot_Clustering_Outliers(mis.datos.numericos, "K-Means Clustering Outliers")
```



Generé un vector booleano indicando para cada valor si es un outlier por el método de clustering o no. Luego setee unos parámetros de forma global para ejecutar la funcion MiBiPlot_Clustering_Outliers que grafica un biplot de los outliers identificados por el método de clustering. Se puede ver marcados con una x los datos que se consideran outliers.

```
mis.datos.medias <- colMeans(mis.datos.numericos)
mis.datos.desviaciones <- apply(X = mis.datos.numericos, MARGIN = 2, FUN = sd)
aux1 <- sweep(centroides.normalizados.pima, 2, mis.datos.desviaciones, FUN="*")
centroides.valores <- sweep(aux1, 2, mis.datos.medias, FUN="+")
centroides.valores
```

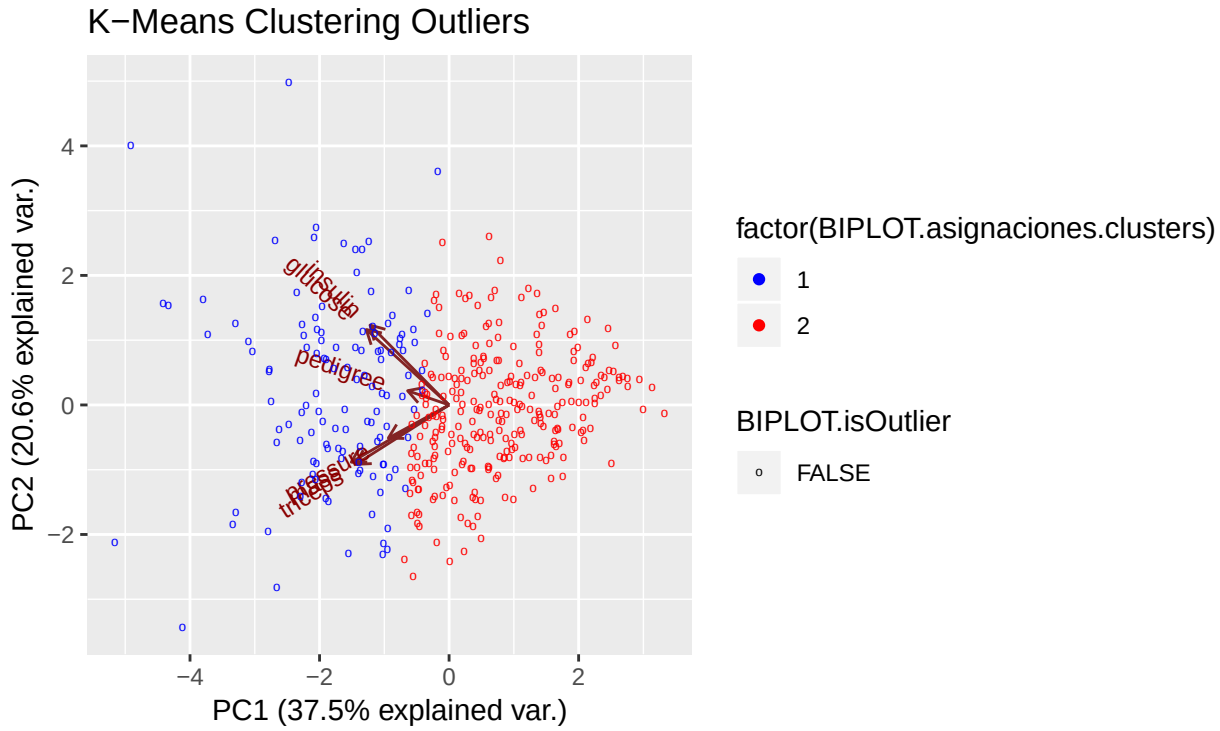
```
##    glucose pressure  triceps  insulin     mass  pedigree
## 1 150.2541 78.11475 37.20492 252.7787 38.34344 0.6546066
## 2 110.1444 67.29630 25.50370 112.3519 30.71074 0.4636000
```

A partir de los datos normalizados, "revertimos" el proceso de normalización para obtener los valores originales de los centroides de los 2 clusters que obtuvimos con k-means.

```
top_clustering_outliers_distancia_mahalanobis = function(datos,
                                                         indices.asignacion.clustering,
                                                         numero.de.outliers){

  cluster.ids = unique(indices.asignacion.clustering)
  k           = length(cluster.ids)
  seleccion   = sapply(1:k, function(x) indices.asignacion.clustering == x)


  # Usando medias y covarianzas:
  # lista.matriz.de.covarianzas   = lapply(1:k, function(x) cov(mis.datos.numericos[seleccion[,x],]))
  # lista.vector.de.medias        = lapply(1:k, function(x) colMeans(mis.datos.numericos[seleccion[,x],


  # Usando la estimaci?n robusta de la media y covarianza: (cov.rob del paquete MASS:
  lista.matriz.de.covarianzas   =
    lapply(1:k, function(x) cov.rob(mis.datos.numericos[seleccion[,x],])$cov)
  lista.vector.de.medias        =
    lapply(1:k, function(x) cov.rob(mis.datos.numericos[seleccion[,x],])$center)


  mah.distances   = lapply(1:k,
                           function(x) mahalanobis(mis.datos.numericos[seleccion[,x],],
                                                   lista.vector.de.medias[[x]],
                                                   lista.matriz.de.covarianzas[[x]]))

  todos.juntos = unlist(mah.distances)
  todos.juntos.ordenados = names(todos.juntos[order(todos.juntos, decreasing=TRUE)])
  indices.top.mah.outliers = as.numeric(todos.juntos.ordenados[1:numero.de.outliers])


  list(distancias = mah.distances[indices.top.mah.outliers]  ,
       indices = indices.top.mah.outliers)
}

top.clustering.outliers.mah = top_clustering_outliers_distancia_mahalanobis(mis.datos.numericos,
                                                                            indices.clustering.pima,
                                                                            numero.de.outliers)
```

```
## Warning in
## top_clustering_outliers_distancia_mahalanobis(mis.datos.numericos, : NAs
## introducidos por coerción
```

```
numero.de.datos = nrow(mis.datos.numericos)
is.kmeans.outlier.mah = rep(FALSE, numero.de.datos)
is.kmeans.outlier.mah[top.clustering.outliers.mah$indices] = TRUE

BIPLOT.isOutlier             = is.kmeans.outlier.mah
BIPLOT.cluster.colors        = c("blue","red","brown")     # Tantos colores como diga numero.de.cluster
BIPLOT.asignaciones.clusters = indices.clustering.pima
MiBiPlot_Clustering_Outliers(mis.datos.numericos, "K-Means Clustering Outliers")
```



En este caso construimos una función que pare el calculo de cada punto a su centroide utiliza la distancia de Mahalanobis.

```
top_clustering_outliers_distancia_relativa = function(datos.normalizados,
                                                      indices.asignacion.clustering,
                                                      datos.centroides.normalizados,
                                                      numero.de.outliers){

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

  cluster.ids = unique(indices.asignacion.clustering)
  k           = length(cluster.ids)

  distancias.a.centroides.por.cluster    =
    sapply(1:k , function(x) dist_centroides [indices.asignacion.clustering  == cluster.ids[x]])
```

```
  distancias.medianas.de.cada.cluster    =
    sapply(1:k , function(x) median(dist_centroides[[x]]))

  todas.las.distancias.medianas.de.cada.cluster  =
    distancias.medianas.de.cada.cluster[indices.asignacion.clustering]
  ratios = dist_centroides   /  todas.las.distancias.medianas.de.cada.cluster

  indices.top.outliers = order(ratios, decreasing=T)[1:numero.de.outliers]

  list(distancias = ratios[indices.top.outliers]  , indices = indices.top.outliers)
}



top.outliers.kmeans.distancia.relativa =
  top_clustering_outliers_distancia_relativa(mis.datos.numericos.normalizados,
    indices.clustering.pima,
    centroides.normalizados.pima,
    numero.de.outliers)


cat("?ndices de los top k clustering outliers (k-means, usando distancia relativa)")
```

```
## ?ndices de los top k clustering outliers (k-means, usando distancia relativa)
```

```
top.outliers.kmeans.distancia.relativa$indices
```

```
## [1] 226 111   2   5  87
```

```
cat("Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa)")
```

```
## Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa)
```

```
top.outliers.kmeans.distancia.relativa$distancias
```

```
##     S226     S111       S2       S5      S87
## 5.664642 5.496311 4.783359 4.746532 4.291260
```