

Trabajo Final Detección de Anomalías

Alberto Armijo Ruiz

28 de enero de 2019

Configuración inicial

Antes de empezar a trabajar, deberemos de establecer el PATH de trabajo y después cargar los ficheros *!Outliers_...* para cargar las librerías y funciones necesarias para trabajar con la práctica. Una vez hemos hecho esto podemos cargar el fichero y comenzar a buscar outliers.

```
# Ponemos el path deseado. PATH_HASTA_DIRECTORIO_TRABAJO_FINAL
setwd(paste("~/Universidad/Mineria\ de\ datos.\ ",  
"Aprendizaje\ no\ supervisado\ ", "y\ deteccion\ de\ anomalias/",  
"trabajo\ final",sep=""))

# Cargamos los ficheros de librerías y funciones.
source("./data/!Outliers_A2_Librerias_a_cargar_en_cada_sesion.R", echo=FALSE)

## Loading required package: plyr
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:reshape':
##       rename, round_any
## Loading required package: scales
## Loading required package: grid
##
## Attaching package: 'EnvStats'
## The following objects are masked from 'package:stats':
##       predict, predict.lm
## The following object is masked from 'package:base':
##       print.default
## Loading required package: sgeostat
## sROC 0.1-2 loaded
## Loading required package: lattice
##
## Attaching package: 'DMwR'
## The following object is masked from 'package:plyr':
##       join
source("./data/!Outliers_A3_Funciones_a_cargar_en_cada_sesion.R", echo=FALSE)
```

Lectura de los datos

Al igual que con el dataset que se utilizó en las sesiones de prácticas, se va a utilizar las mismas variables cambiando solamente sus valores. El dataset que vamos a utilizar se puede obtener de la siguiente página <https://sci2s.ugr.es/keel/dataset.php?cod=102>.

```
# leemos los datos
mydata.numeric = read.csv('./data/magic.dat', header=FALSE, comment.char = "@")

# añadimos nombres de las columnas
names(mydata.numeric) = c('FLength', 'FWidth', 'FSize', 'FConc', 'FConc1', 'FAsym', 'FM3Long', 'FM3Trans')

# dado que es un dataset de clasificación, quitaremos para el estudio de anomalías la variable 'Class'.
mydata.numeric = subset(mydata.numeric, select=-Class)

head(mydata.numeric)

##      FLength    FWWidth   FSize   FConc FConc1     FAsym   FM3Long FM3Trans
## 1 28.7967 16.0021 2.6449 0.3918 0.1982 27.7004 22.0110 -8.2027
## 2 31.6036 11.7235 2.5185 0.5303 0.3773 26.2722 23.8238 -9.9574
## 3 162.0520 136.0310 4.0612 0.0374 0.0187 116.7410 -64.8580 -45.2160
## 4 23.8172  9.5728 2.3385 0.6147 0.3922 27.2107 -6.4633 -7.1513
## 5 75.1362 30.9205 3.1611 0.3168 0.1832 -5.5277 28.5525 21.8393
## 6 51.6240 21.1502 2.9085 0.2420 0.1340 50.8761 43.1887  9.8145
##      FAlpha     FDist
## 1 40.0920 81.8828
## 2  6.3609 205.2610
## 3 76.9600 256.7880
## 4 10.4490 116.7370
## 5  4.6480 356.4620
## 6  3.6130 238.0980

# declaramos el resto de variables
indice.columna = 1
nombre.mydata = "magic"

# declaramos valores escalados. Necesarios para algunos apartados de la práctica.
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]
```

Ahora ya podemos comenzar con el estudio de anomalías.

Cómputo de Outliers IQR

Lo primero que haremos será calcular el IQR de la columna que hemos seleccionado, después analizaremos qué datos son anomalías según el análisis de ese IQR.

```
cuartil.primero = quantile(columna.scaled, 0.25)
cuartil.tercero = quantile(columna.scaled, 0.75)
iqr = IQR(columna.scaled)

# Ahora calculamos los valores para calcular los outliers.
```

```

extremo.superior.outlier.normal = cuartil.tercero + (1.5*iqr)
extremo.inferior.outlier.normal = cuartil.primer - (1.5*iqr)
extremo.inferior.outlier.extremo = cuartil.primer - (3*iqr)
extremo.superior.outlier.extremo = cuartil.tercero + (3*iqr)

```

Ahora calculamos los vectores.

```

vector.es.outlier.normal = columna.scaled < extremo.inferior.outlier.normal | columna.scaled > extremo.
vector.es.outlier.extremo = columna.scaled < extremo.inferior.outlier.extremo | columna.scaled > extremo.

```

Dado que tenemos una gran cantidad de datos, para el caso de este dataset son 19020 instancias, miraremos si los vectores que hemos calculado anteriormente contienen algún dato igual a *TRUE*.

```

cat("outliers normales:", any(vector.es.outlier.normal==TRUE), "\n",
    "outliers extremos:", any(vector.es.outlier.extremo==TRUE))

```

```

## outliers normales: TRUE
## outliers extremos: TRUE

```

Por lo que podemos ver en la salida, nuestro dataset contiene tanto outliers normales ($< o > 1.5 * IQR$) como outliers extremos ($< o > 3 * IQR$). Lo siguiente que vamos a hacer es calcular y mostrar estos datos que son outliers.

Valores outliers normales

```

claves.outliers.normales = which(vector.es.outlier.normal)
head(claves.outliers.normales)

```

```

##      3   991  1431  1896  2448  2575
##      3   991  1431  1896  2448  2575

```

```

data.frame.outliers.normales = mydata.numeric[claves.outliers.normales,]
head(data.frame.outliers.normales)

```

```

##      FLength   FWidth   FSize   FConc   FConc1   FAsym   FM3Long   FM3Trans
## 3     162.052 136.0310 4.0612 0.0374 0.0187  116.741 -64.8580 -45.2160
## 991   138.838 32.8249 3.2464 0.1713 0.1035 -122.634  95.0450 -27.8132
## 1431  161.278 18.5719 2.9159 0.2330 0.1220  107.855 162.4600 14.6187
## 1896  141.884 18.1945 2.5459 0.4154 0.2319 -146.974  98.6042 -11.7328
## 2448  147.383 35.4870 2.9256 0.3169 0.2178 -168.736 -57.8306 -28.5496
## 2575  158.860 29.3982 2.8373 0.2953 0.1855 -104.699 -110.1540 -6.6452
##      FAlpha   FDist
## 3     76.9600 256.788
## 991   3.4990 297.614
## 1431  5.6891 225.220
## 1896  5.7040 310.081
## 2448  32.7830 227.050
## 2575  0.0103 353.410

```

```

nombres.outliers.normales = row.names(data.frame.outliers.normales)
head(nombres.outliers.normales)

```

```

## [1] "3"    "991"  "1431" "1896" "2448" "2575"

```

```

valores.outliers.normales = data.frame.outliers.normales[,indice.columna]
head(valores.outliers.normales)

```

```

## [1] 162.052 138.838 161.278 141.884 147.383 158.860

```

Valores outliers extremos

```

claves.outliers.extremos = which(vector.es.outlier.extremo)

```

```

head(claves.outliers.extremos)

## 2795 5838 12338 12414 12459 12499
## 2795 5838 12338 12414 12459 12499

data.frame.outliers.extremos = mydata.numeric[claves.outliers.extremos,]
head(data.frame.outliers.extremos)

##      FLength   FWidth   FSize   FConc FConc1     FAsym   FM3Long FM3Trans
## 2795 272.0630 20.1242 2.5563 0.4556 0.2319 -349.7570 203.8630 -13.8784
## 5838 229.8170 9.5128 2.2934 0.4733 0.2672 -163.4190 -183.3050 7.5867
## 12338 215.3230 67.8238 3.4396 0.1363 0.0725 298.6140 -95.0773 -57.2209
## 12414 207.6170 62.3419 3.5013 0.1148 0.0601 -123.7480 -138.9220 -50.4269
## 12459 297.1239 111.0237 4.0798 0.0928 0.0429 -291.9184 -269.2064 87.1154
## 12499 255.7490 83.1878 4.0608 0.0668 0.0380 -224.0950 -157.1630 -62.1765
##      FAlpha    FDist
## 2795 62.3504 184.0600
## 5838 73.9750 118.2470
## 12338 76.9240 243.0640
## 12414 6.3810 218.1300
## 12459 41.3357 232.9571
## 12499 76.7151 276.2450

nombres.outliers.extremos = row.names(data.frame.outliers.extremos)
head(nombres.outliers.extremos)

## [1] "2795"  "5838"  "12338" "12414" "12459" "12499"

valores.outliers.extremos = data.frame.outliers.extremos[,indice.columna]
head(valores.outliers.extremos)

## [1] 272.0630 229.8170 215.3230 207.6170 297.1239 255.7490

# Mostramos dichos valores.
cat("indices de los outliers normales:\n",
    nombres.outliers.normales, "\n",
    "valores de los outliers:\n",
    valores.outliers.normales, "\n",
    "número de outliers normales:", length(valores.outliers.normales), "\n")

## indices de los outliers normales:
## 3 991 1431 1896 2448 2575 2645 2795 3299 3361 3866 4301 4710 4777 5260 5413 5615 5838 6805 7677 770
## valores de los outliers:
## 162.052 138.838 161.278 141.884 147.383 158.86 150.978 272.063 172.979 142.563 196.051 141.235 140.3
## número de outliers normales: 971

cat("indices de los outliers extremos:\n",
    nombres.outliers.extremos, "\n",
    "valores de los outliers:\n",
    valores.outliers.extremos, "\n",
    "número de outliers extremos:", length(valores.outliers.extremos), "\n")

## indices de los outliers extremos:
## 2795 5838 12338 12414 12459 12499 12508 12510 12528 12541 12566 12629 12637 12658 12664 12705 12724
## valores de los outliers:
## 272.063 229.817 215.323 207.617 297.1239 255.749 240.462 212.881 229.285 261.9504 258.5767 213.6676
## número de outliers extremos: 236

```

Como se puede ver en los resultados, tenemos 971 datos en la columna que se consideran outliers normales, de esos, 236 datos también son outliers extremos. Ahora obtendremos los valores de dichos outliers y veremos la desviación con respecto a la media de la columna.

```
valores.normalizados.outliers.normales = columna.scaled[claves.outliers.normales]
sd(valores.normalizados.outliers.normales)
```

```
## [1] 0.8728221
```

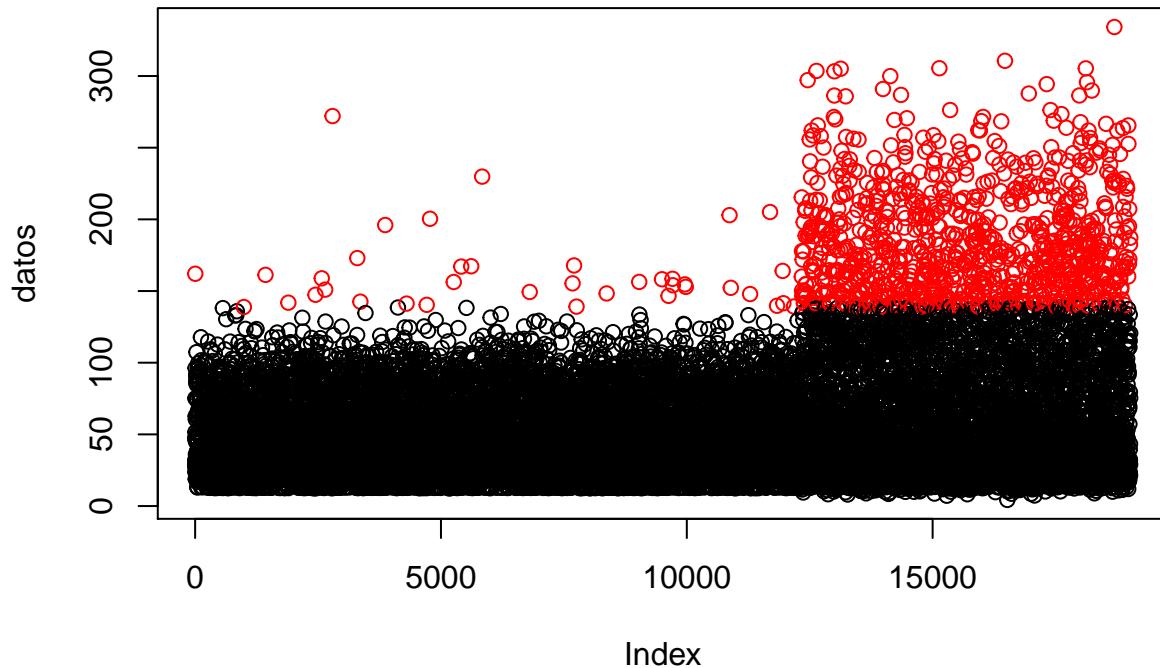
```
valores.normalizados.outliers.extremos = columna.scaled[claves.outliers.extremos]
sd(valores.normalizados.outliers.extremos)
```

```
## [1] 0.5845424
```

El siguiente paso será mostrar en un gráfico los datos que son outliers para identificarlos mejor, para ello haremos uso de la función *MiPlot_Univariate_Outliers()*.

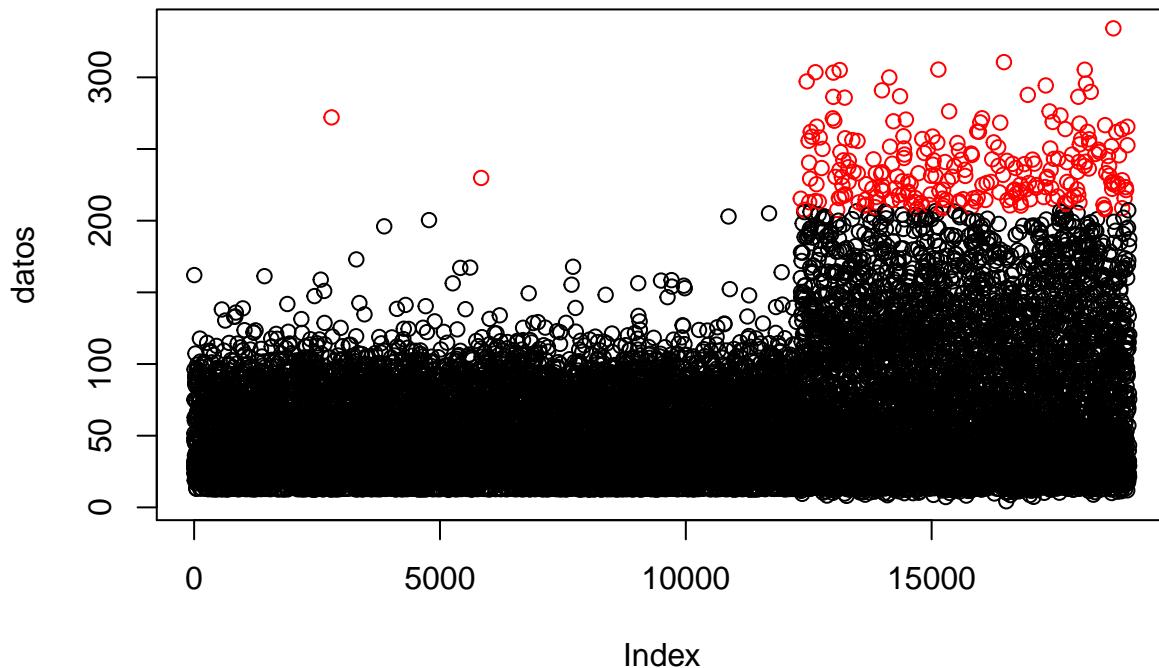
```
MiPlot_Univariate_Outliers(columna,claves.outliers.normales, "Outliers normales")
```

Outliers normales



```
MiPlot_Univariate_Outliers(columna,claves.outliers.extremos, "Outliers extremos")
```

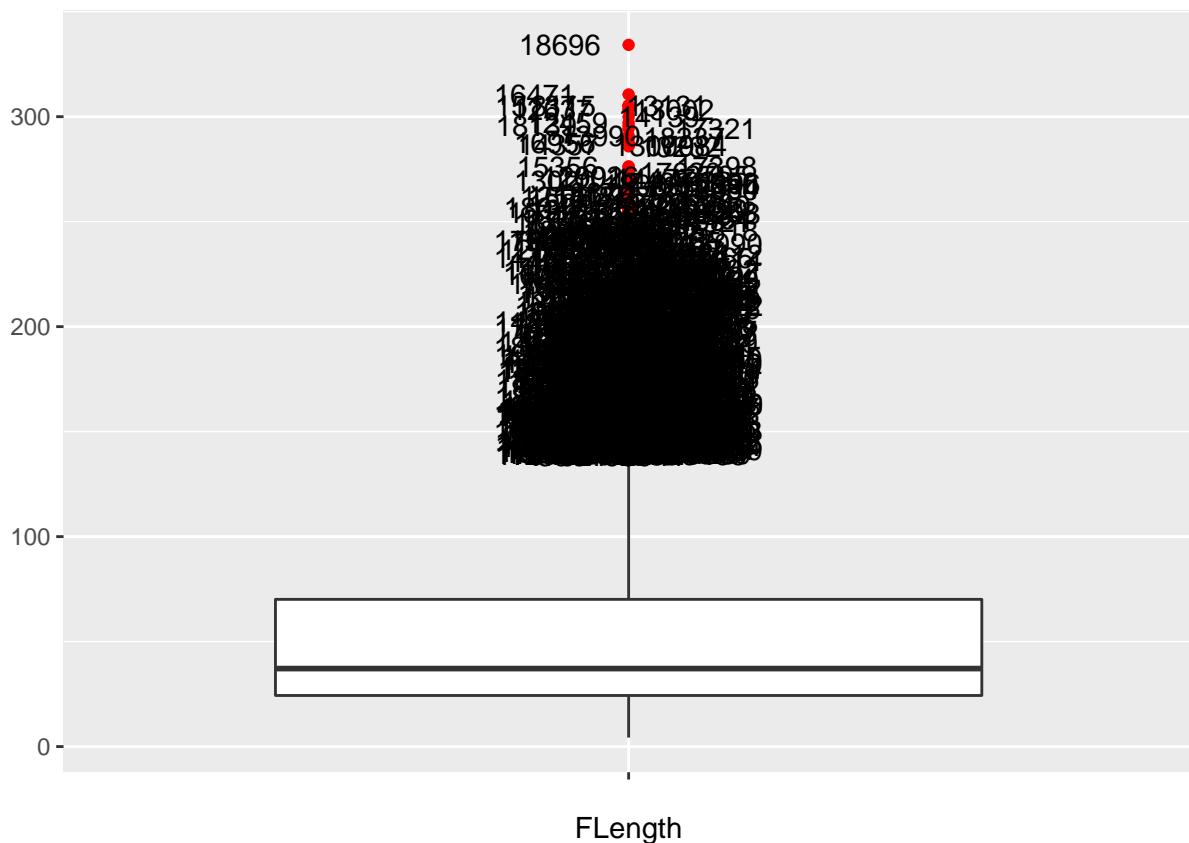
Outliers extremos



Gracias a la información que se muestra en los gráficos, podemos que nuestros outliers en su gran mayoría se encuentran al final del dataset (más o menos en el último tercio de los datos) menos unos pocos. Esto puede hacernos pensar que sea posible que esa gran cantidad de datos anómalos esté relacionada también con alguna otra variable del dataset. Otro tipo de gráfica que usaremos es un Boxplot. Para ello utilizaremos la función *MiBoxPlot_IQR_Univariate_Outliers()*.

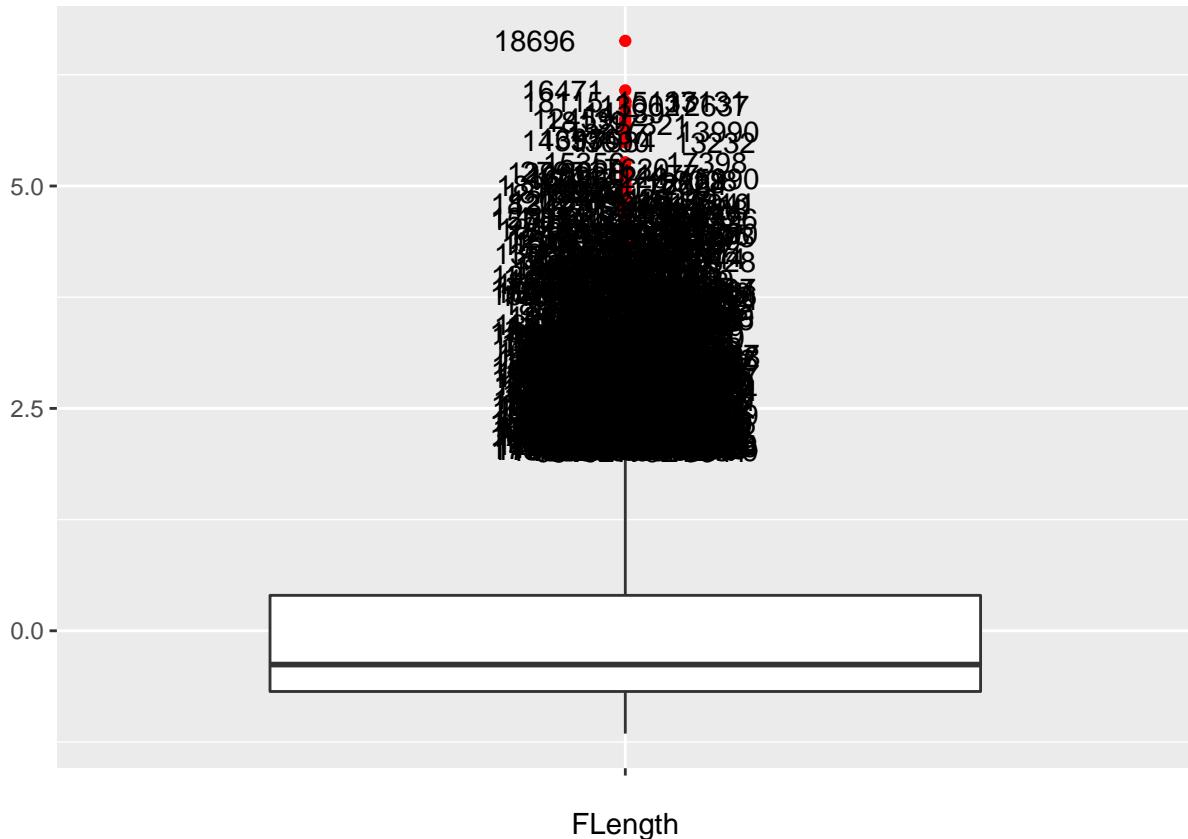
```
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric,indice.columna,coef=1.5)
```

```
## [1] 971
```



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

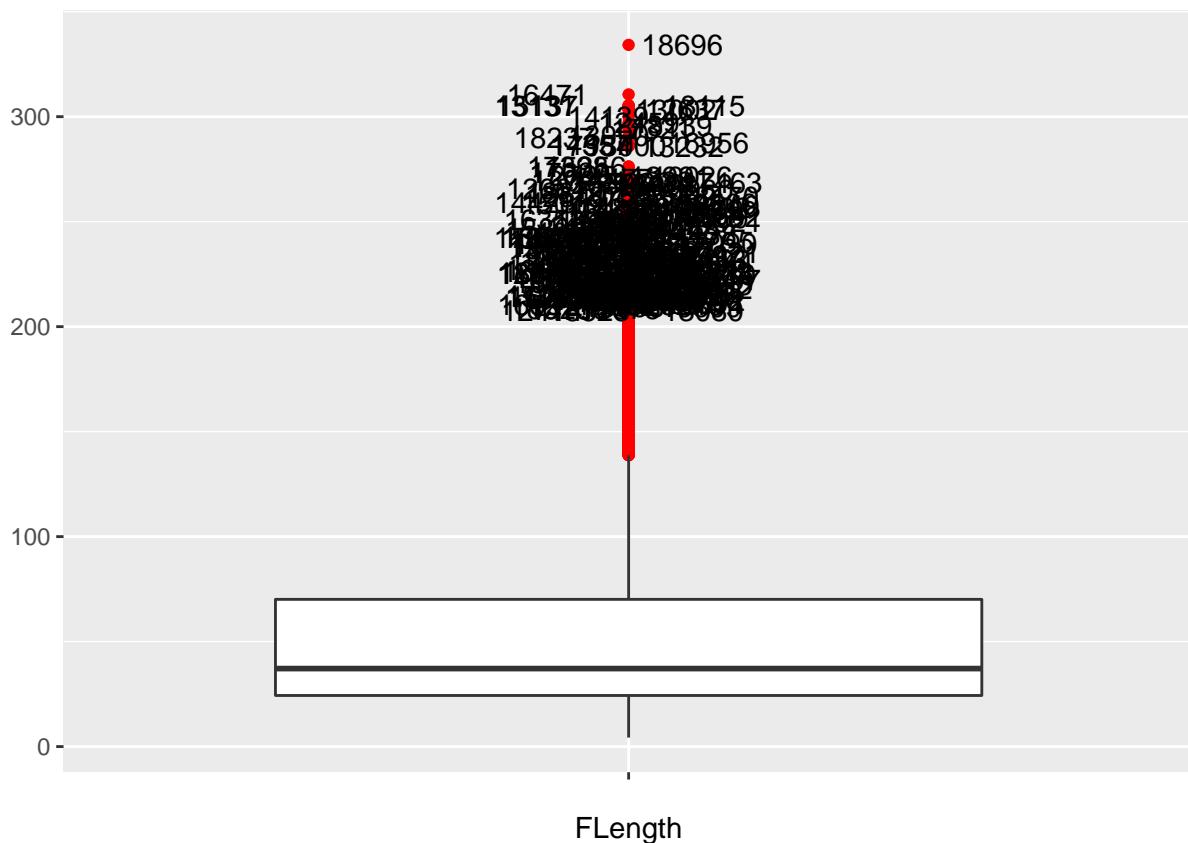
```
## [1] 971
```



En este caso solamente lo utilizamos para los outliers normales, ya que la función de ggplot no nos permite especificar un valor para calcular el outlier, utiliza directamente $1.5 * IQR$. Si quisieramos ver donde están representados los datos , podríamos mostrar los labels de las instancias.

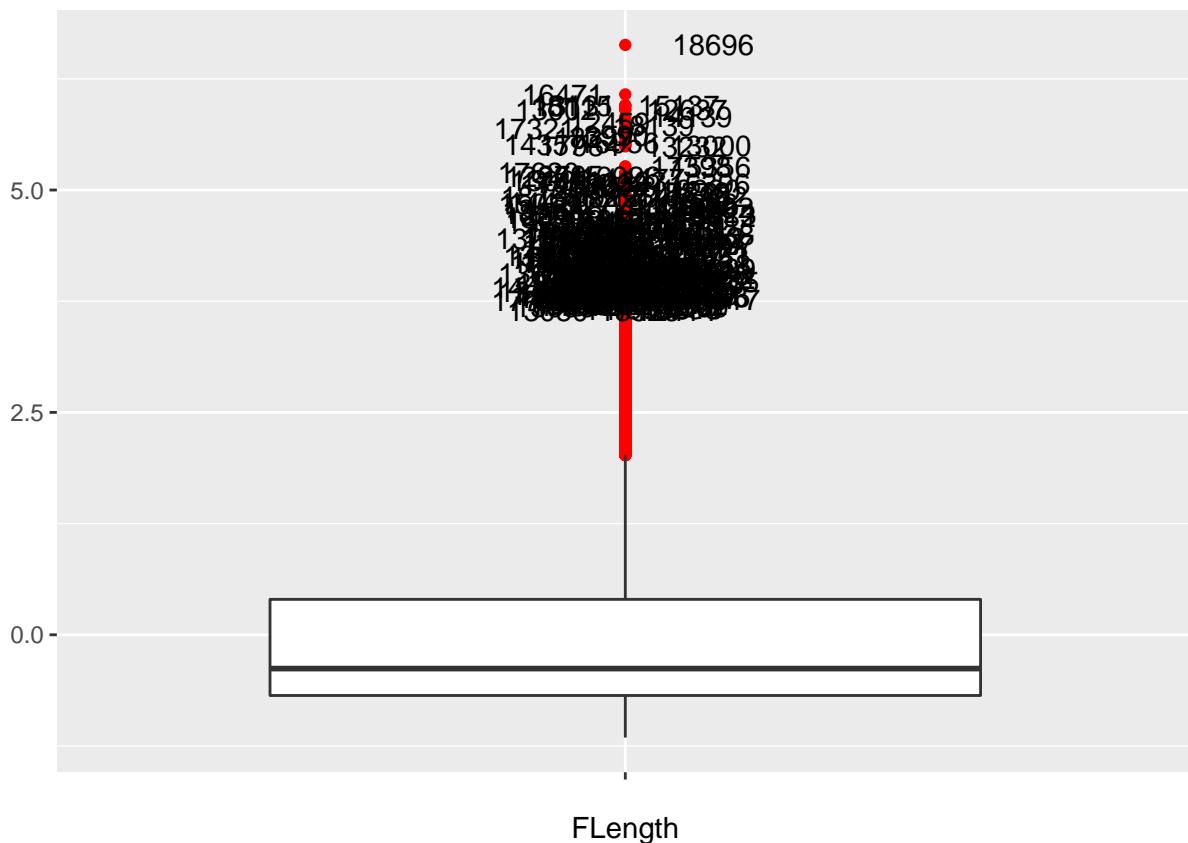
```
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric,indice.columna,coef=3)
```

```
## [1] 236
```



```
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric.scaled,indice.columna,coef=3)
```

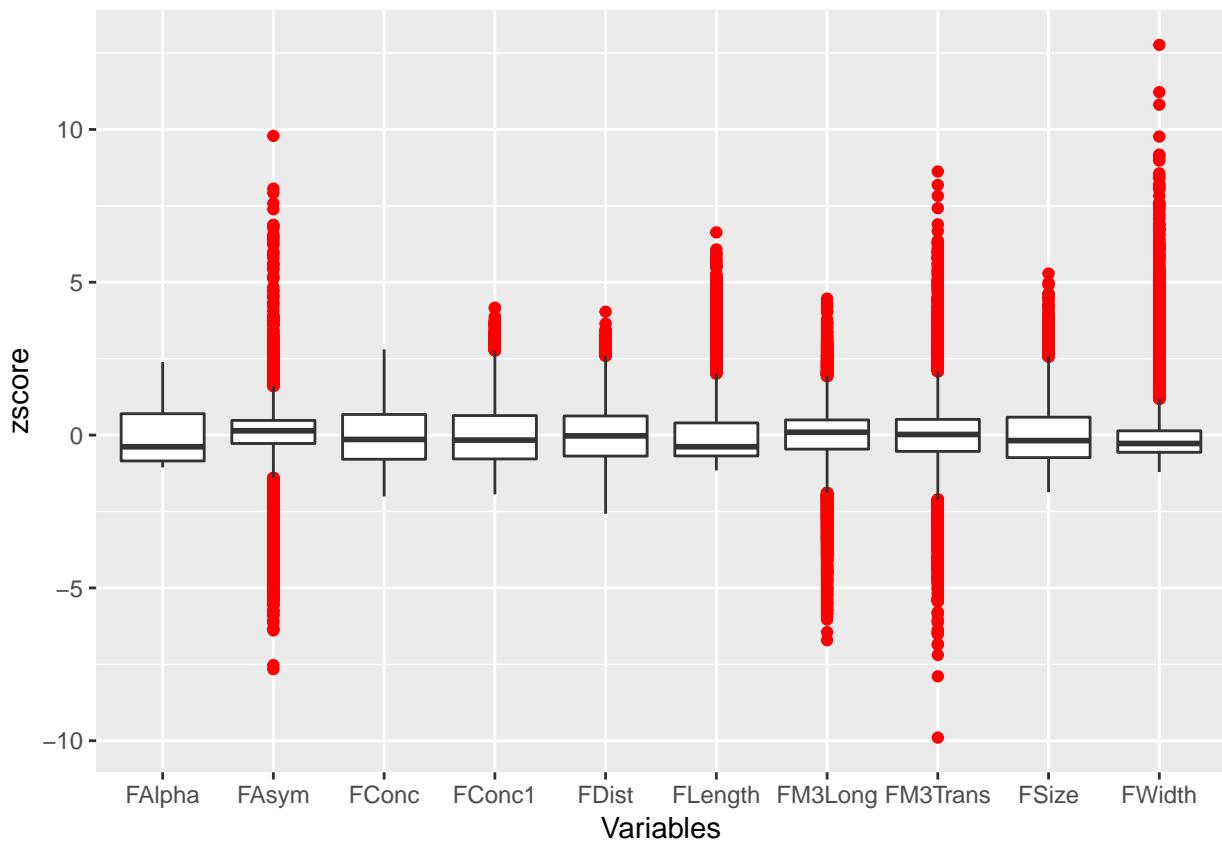
```
## [1] 236
```



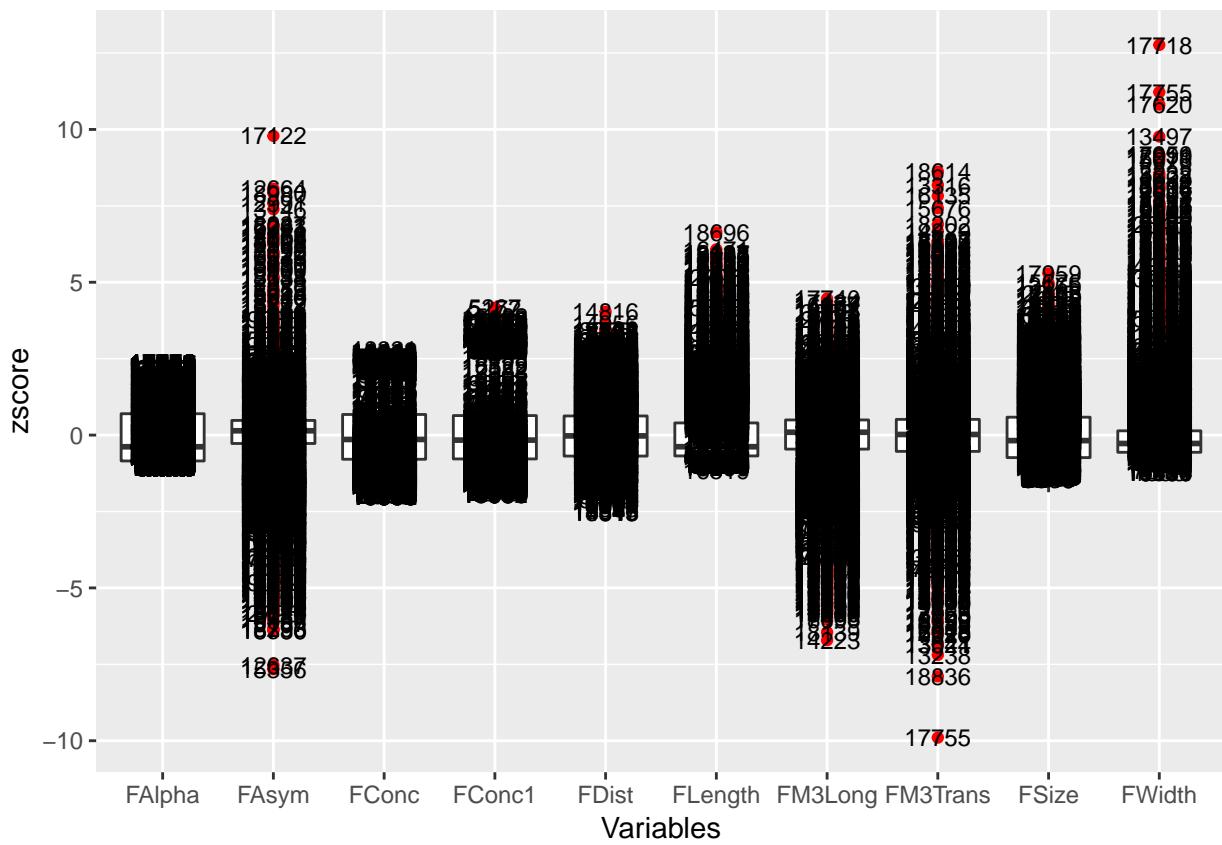
Como se puede ver, los outliers que se consideran extremos son aquellos que tiene valores superiores a 200 para la columna que estamos analizando.

Lo siguiente que vamos a hacer es mirar que valores toman cada outlier en cada una de las columnas de nuestro dataset, así podemos apreciar si las instancias que toman valores anómalos en la columna que hemos estado estudiando toman también valores anómalos para otras columnas; para ello haremos uso de la función *MiBoxPlot_juntos()* y *MiBoxPlot_juntos_con_etiquetas()*.

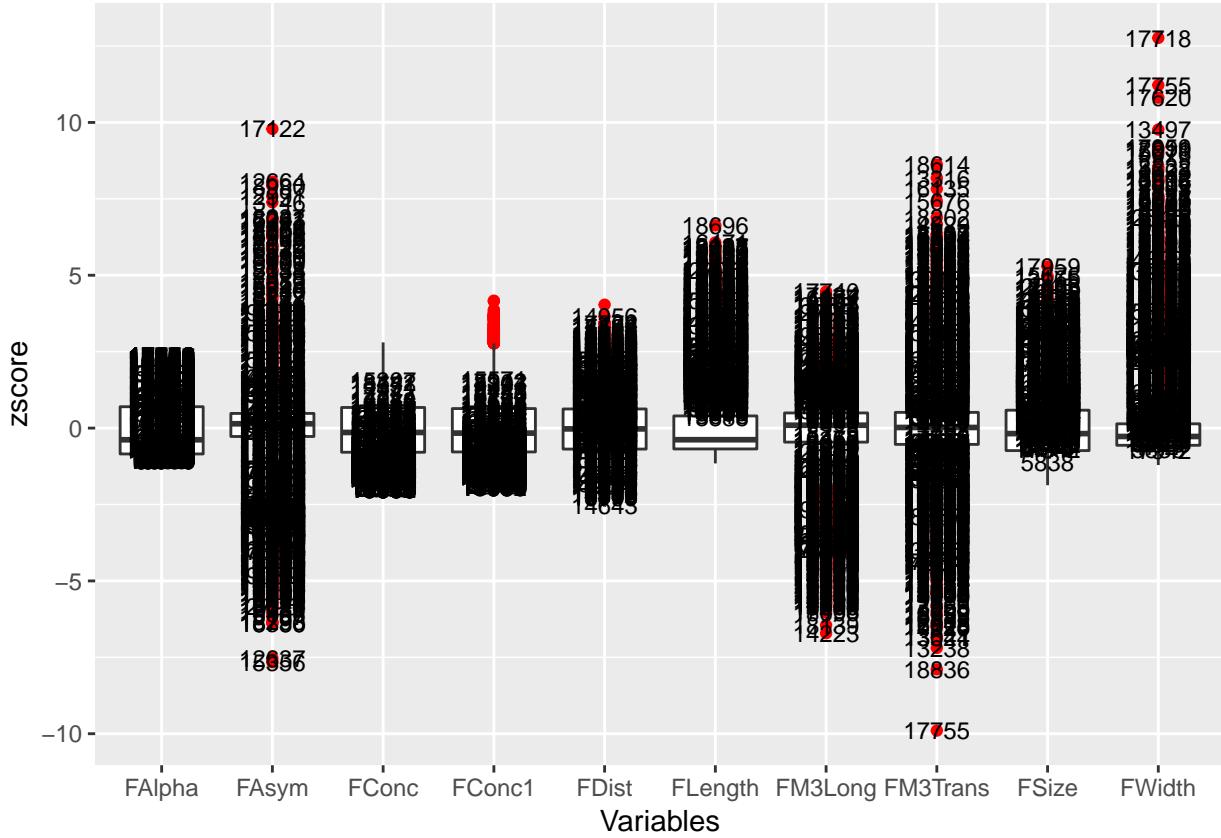
```
# BoxPlot para outliers normales
MiBoxPlot_juntos(mydata.numeric)
```



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



```
# BoxPlot para outliers extremos
MiBoxPlot_juntos_con_etiquetas(mydata.numeric,coef=3.0)
```



Como se puede ver en las gráficas, la gran mayoría de las variables contienen tanto outliers normales como extremos. Por desgracia al haber tantos datos considerados como outliers es difícil saber si un label aparece en dos columnas distintas, aún así es posible comprobarlo. Lo que sí se puede apreciar, al menos por lo poco que se vé en las etiquetas, es que una buena parte de los datos que considera como outliers son aquellos que se encuentran al final del dataset.

Ahora lo que vamos a hacer es calcular las filas del dataset que contienen un outlier en algunas de sus columnas, para ello vamos a utilizar la función `vector_claves_outliers_IQR()`; a la cual hay que pasarle nuestro conjunto de datos, la columna para la cual queremos calcular las claves y el coeficiente con el cual calcula si es outlier o no (por defecto es 1.5).

```
v = 1:ncol(mydata.numeric)
indices.en.alguna.columna = sapply( v, vector_claves_outliers_IQR, datos=mydata.numeric)
indices.en.alguna.columna = unlist(indices.en.alguna.columna)
indices.en.alguna.columna = sort(unique(indices.en.alguna.columna))
head(indices.en.alguna.columna,10)

## [1] 3 9 21 53 59 86 106 115 118 123

v = 1:ncol(mydata.numeric)
indices.en.alguna.columna.extremos = sapply( v, vector_claves_outliers_IQR, datos=mydata.numeric, coef=3)
indices.en.alguna.columna.extremos = unlist(indices.en.alguna.columna.extremos)
indices.en.alguna.columna.extremos = sort(unique(indices.en.alguna.columna.extremos))
head(indices.en.alguna.columna.extremos,10)

## [1] 3 21 124 564 577 838 849 1184 1496 1569

cat("número de outliers en alguna columna:",
    length(indices.en.alguna.columna),"\n",
```

```

"número de outliers extremos en alguna columna:",
length(indices.en.alguna.columna.extremos), "\n")

## número de outliers en alguna columna: 3018
## número de outliers extremos en alguna columna: 1219

Como se puede ver, el número de outliers normales es mucho mayor que el número de outliers extremos.
Ahora lo que haremos será crear un dataset para cada tipo de outlier con los datos normalizados.

mis.datos.outliers.normalizados = mydata.numeric.scaled[indices.en.alguna.columna,]
head(mis.datos.outliers.normalizados)

##      FLength      FWidth      FSize      FConc      FConc1      FAsym
## 3  2.5682100 6.205695216  2.6157143 -1.8758338 -1.7731944  2.0449383
## 9  1.0145803 1.326472207  2.8120754 -1.6542961 -1.5895020  1.9370777
## 21 0.9062263 2.709074527  2.1654379 -1.6537491 -1.5533064 -0.8846891
## 53 0.2857851 0.009120967 -0.1849708 -0.5187103 -0.4936815 -1.4349705
## 59 0.9977739 0.672211705  2.7602344 -1.5306726 -1.4510540  0.4312151
## 86 0.8104252 0.610116616  2.3186335 -1.3966560 -1.3388478  1.7488892
##      FM3Long      FM3Trans      FAlpha      FDist
## 3   -1.4784975 -2.1829725  1.8891745  0.8426130
## 9    1.4608408  2.0614476 -0.8731243  0.7280433
## 21   1.2109002  2.8697371 -0.4223057  1.2256628
## 53  -0.8194088 -0.9606763 -0.8723582  1.0552668
## 59   1.5365936 -0.7494453 -1.0558576  0.8973554
## 86   1.2641511 -0.7026849 -0.9660616  0.8229025

mis.datos.outliers.normalizados.extremos = mydata.numeric.scaled[indices.en.alguna.columna.extremos,]
head(mis.datos.outliers.normalizados.extremos)

##      FLength      FWidth      FSize      FConc      FConc1      FAsym      FM3Long
## 3  2.5682100 6.2056952  2.6157143 -1.875834 -1.773194  2.0449383 -1.478497
## 21 0.9062263 2.7090745  2.1654379 -1.653749 -1.553306 -0.8846891  1.210900
## 124 1.1437038 0.9466958  2.3713209 -1.280691 -1.245644 -2.5933198  1.173840
## 564 2.0024817 1.5457945  2.4307794 -1.641168 -1.575024 -2.9510704  1.273708
## 577 0.5704645 0.2317792  0.5975113 -1.110572 -1.090908 -2.6835471  1.197034
## 838 1.8848842 3.5814309  3.1193129 -1.775732 -1.679991  1.9964973  1.921259
##      FM3Trans      FAlpha      FDist
## 3   -2.182972  1.8891745  0.8426130
## 21   2.869737 -0.4223057  1.2256628
## 124  1.051530 -0.9032466  1.7854915
## 564  1.487013 -0.8318657  0.8152618
## 577  0.817286 -1.0070904  0.8605839
## 838 -3.075353 -0.7262826 -0.2535337

```

Para esto que hemos hecho antes se podría haber creado una función como la siguiente.

```

vector_claves_outliers_IQR_en_alguna_columna = function(datos,coef=1.5){
  v = 1:ncol(datos)
  indices.en.alguna.columna = sapply( v, vector_claves_outliers_IQR,datos=datos,coef=coef)
  indices.en.alguna.columna = sort(unique(unlist(indices.en.alguna.columna)))
  indices.en.alguna.columna
}

head(vector_claves_outliers_IQR_en_alguna_columna(mydata.numeric))

## [1] 3 9 21 53 59 86

```

Con la función anterior, podemos crear una función que devuelva un vector con aquellas instancias que contienen algún índice, dicha función sería la siguiente.

```
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)  
}  
  
head(vector_es_outlier_IQR_en_alguna_columna(mydata.numeric),10)  
  
## [1] FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

Detección de outliers con test estadísticos

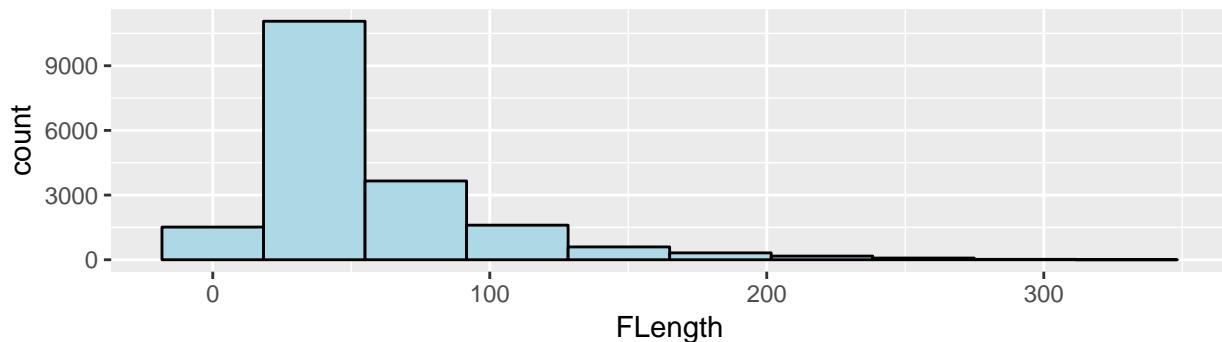
En este apartado realizaremos la detección de outliers en nuestro dataset con test estadísticos, como por ejemplo el test de Grubbs.

Comenzaremos mostrando los datos para ver si se puede identificar algún dato a simple vista.

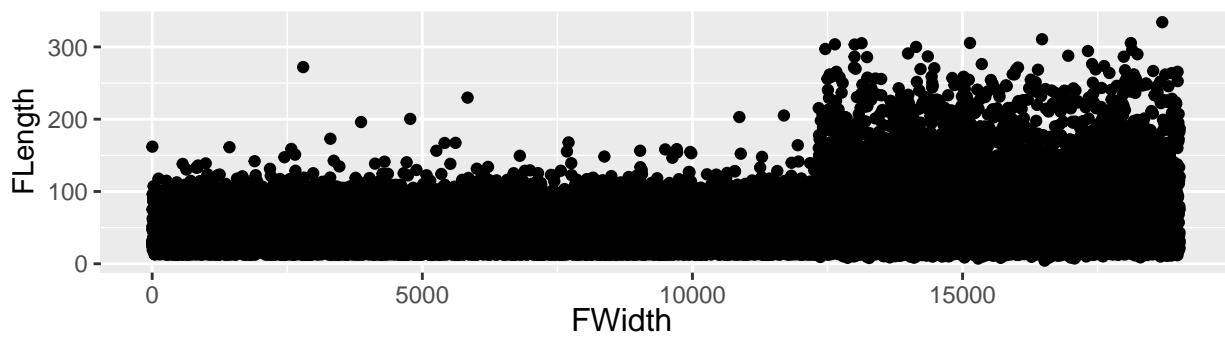
```
histogram_by = function(datos,var, bins=5){  
  
  ggplot(datos,aes_string(x=var)) +  
    geom_histogram(fill='lightblue', color="black", bins=bins)+  
    ggtitle("histogram")  
}  
  
scatter_by = function(datos,var){  
  x= seq_along(datos[,var])  
  ggplot(datos,aes_string(y=var,x=x))+  
    geom_point() + xlab("") + ggtitle("scatter")  
}  
  
hist_and_scatter=function(datos,var,bins=10){  
  h = histogram_by(datos,var,bins)  
  s = scatter_by(datos,var)  
  gridExtra::grid.arrange(h,s,nrow=2,  
                         top=var)  
}  
  
lapply(names(mydata.numeric),hist_and_scatter,datos=mydata.numeric)
```

FLength

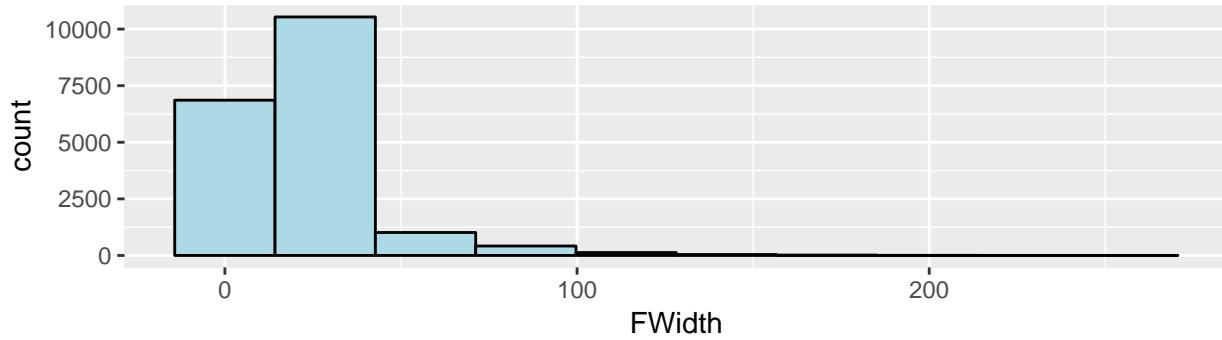
histogram



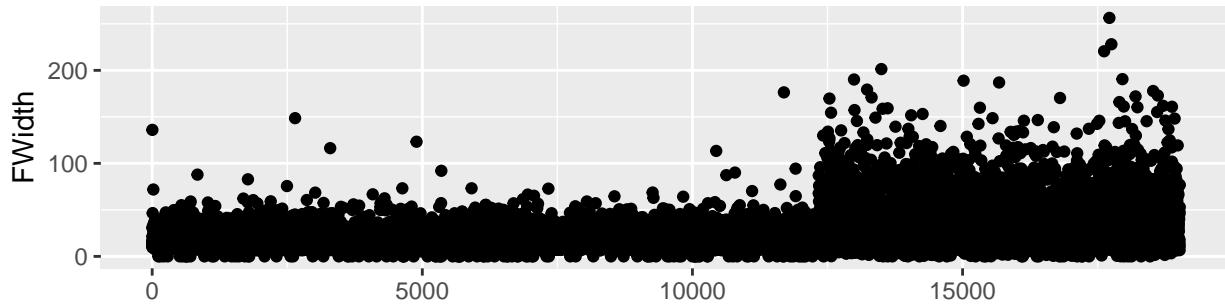
scatter



histogram

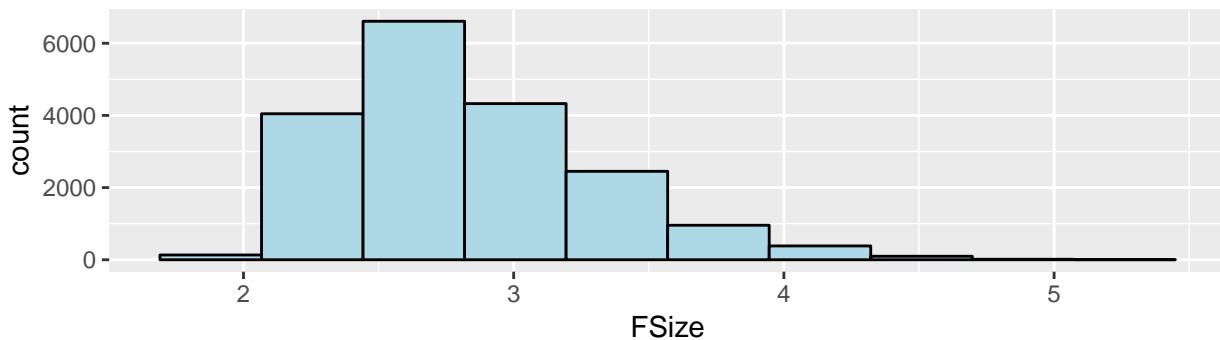


scatter

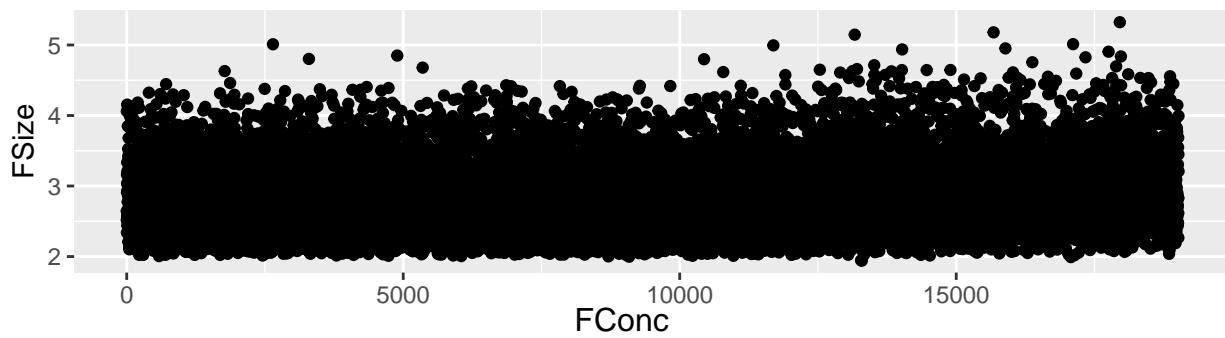


FSize

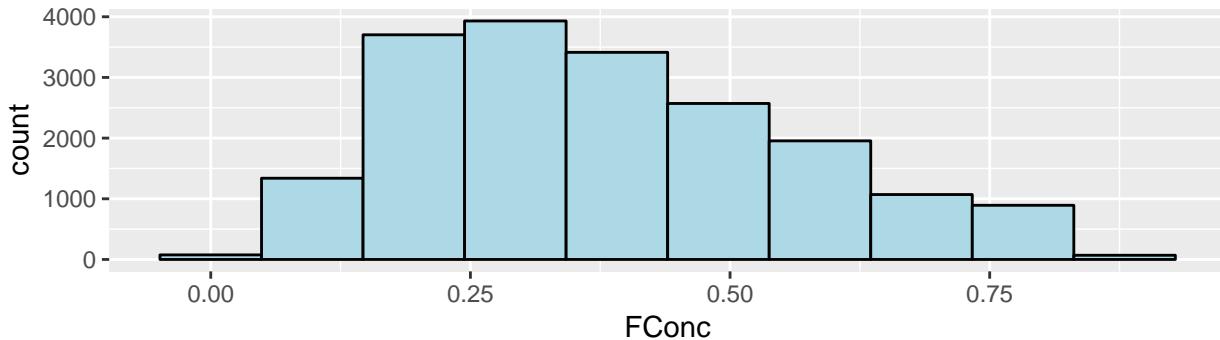
histogram



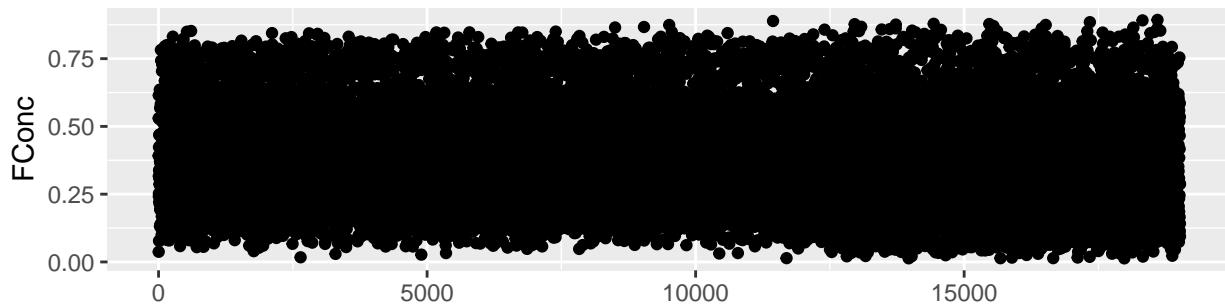
scatter



histogram

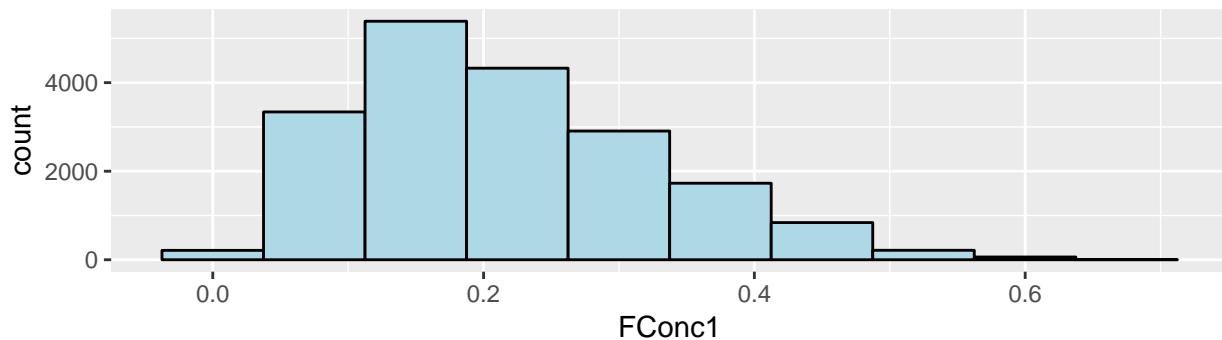


scatter

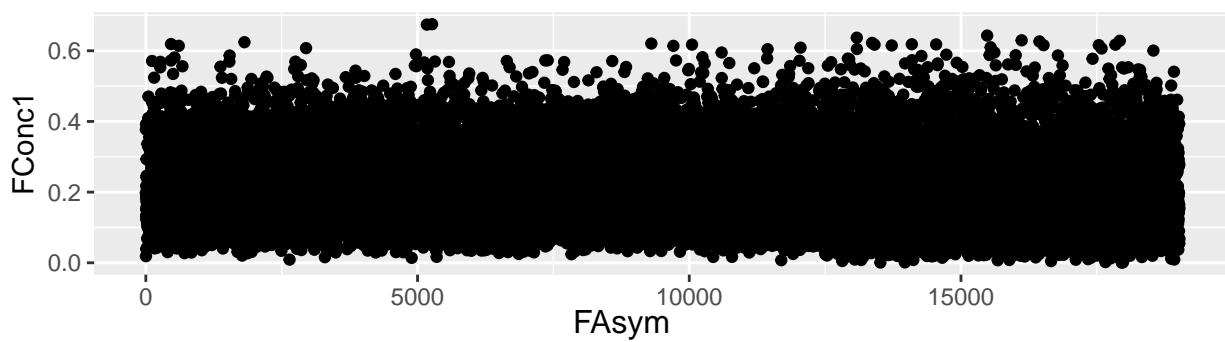


FConc1

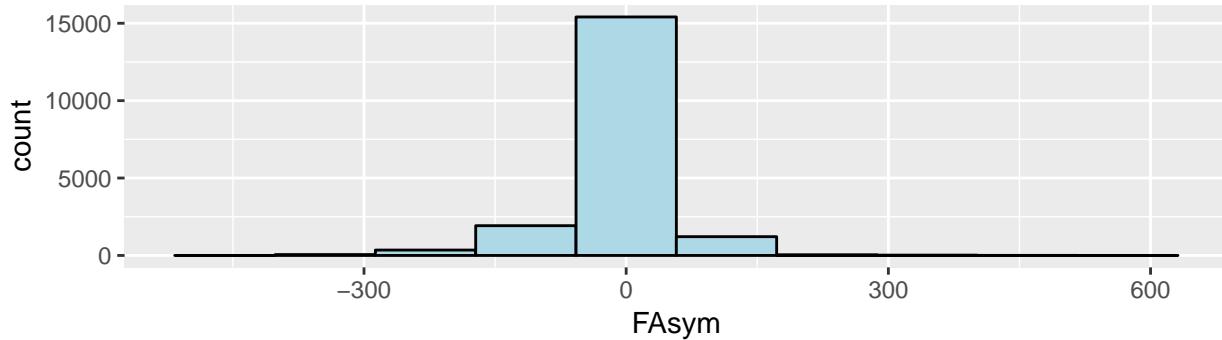
histogram



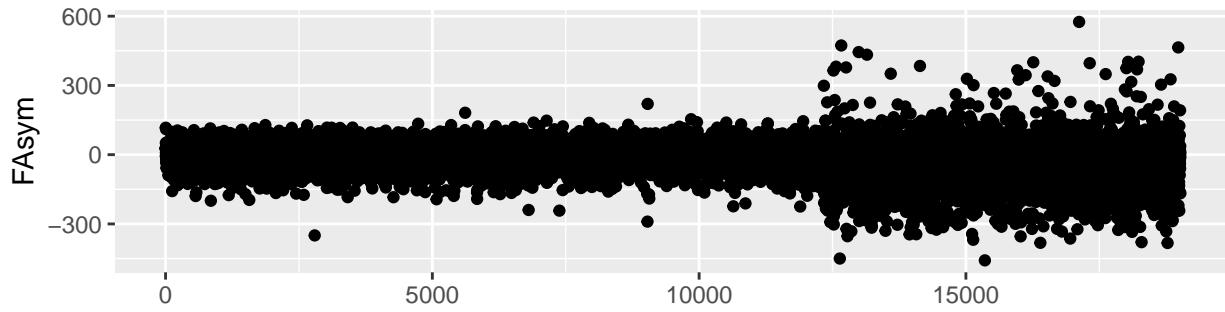
scatter



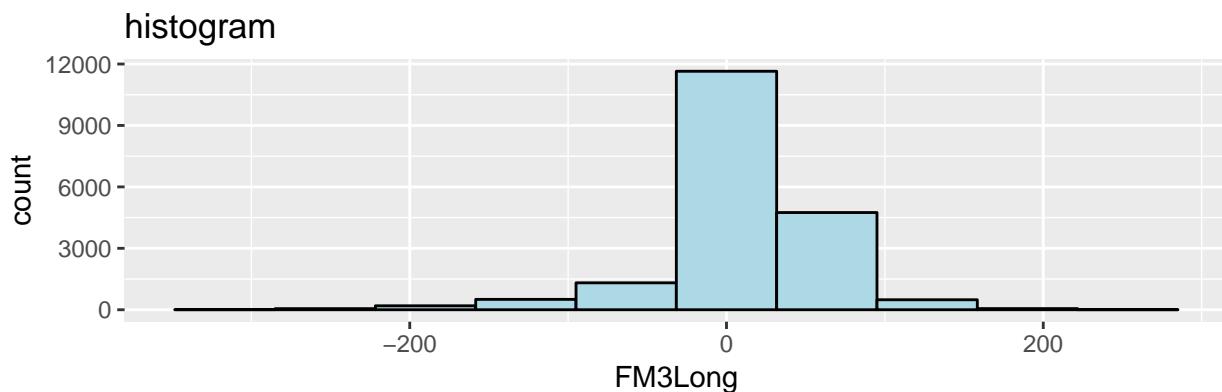
histogram



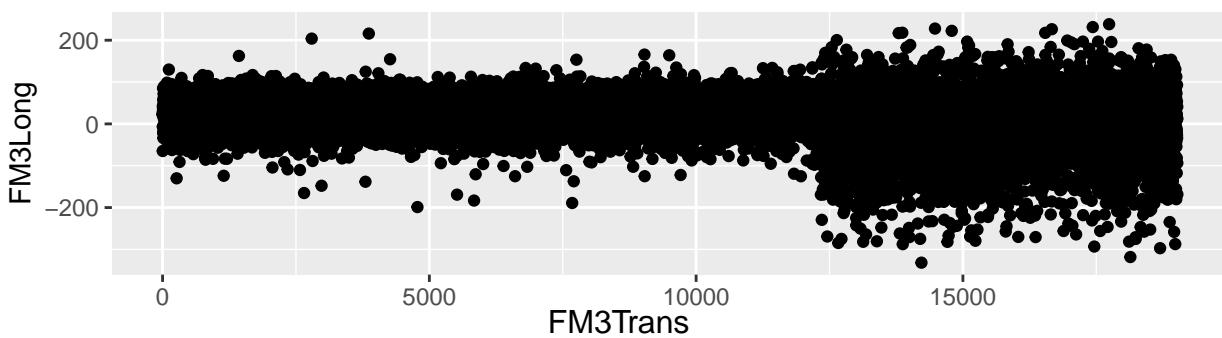
scatter



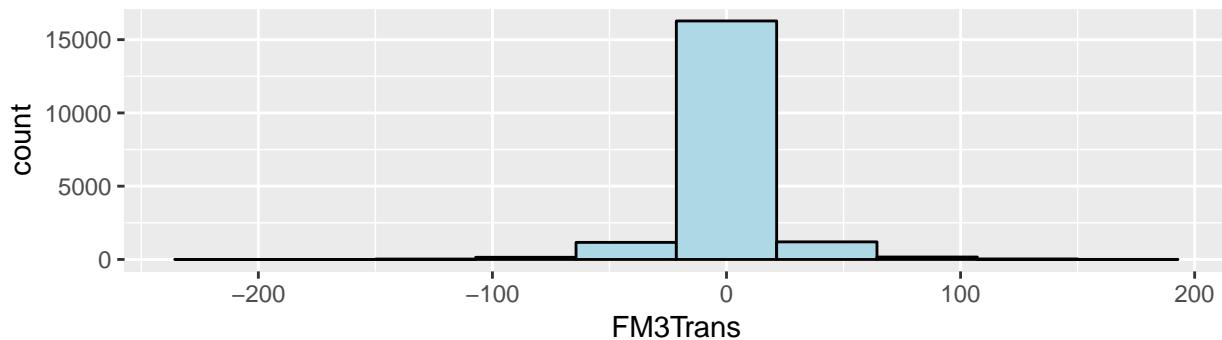
FM3Long



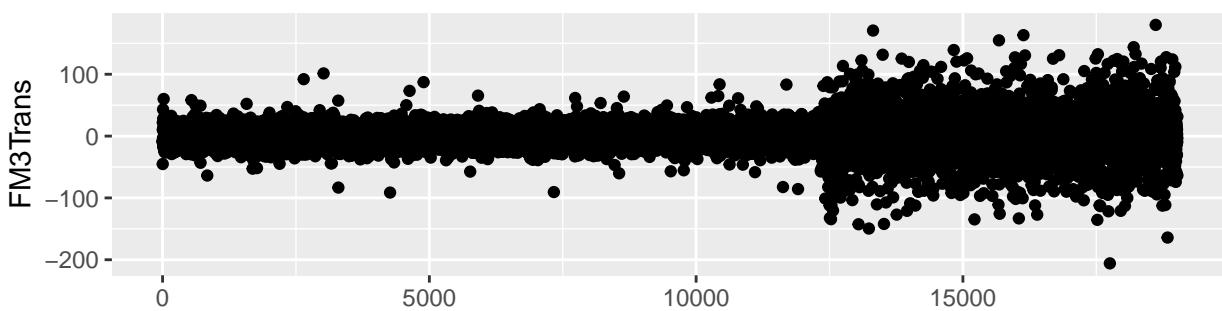
scatter



histogram

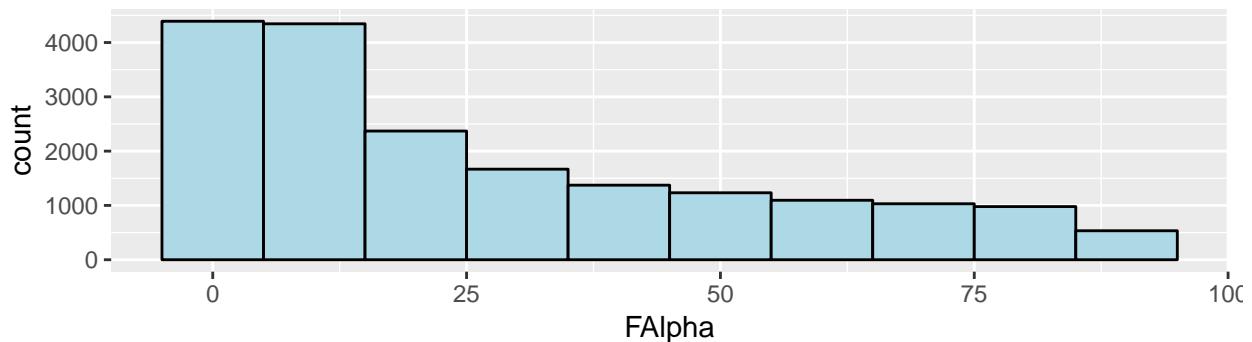


scatter

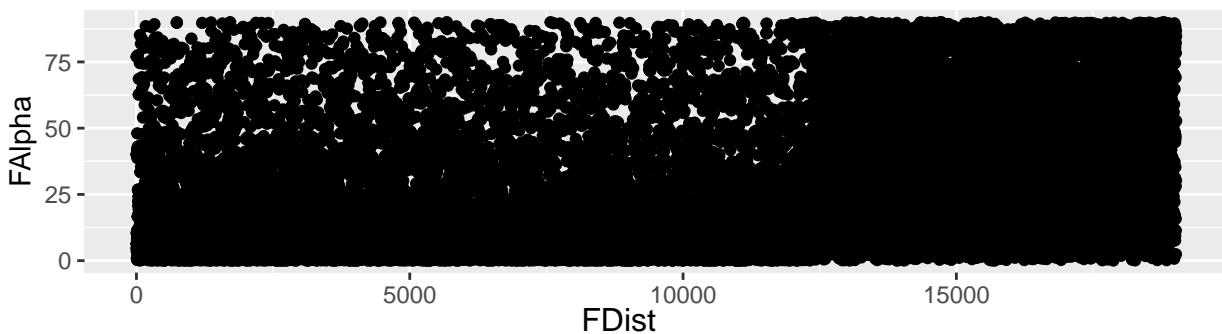


FAlpha

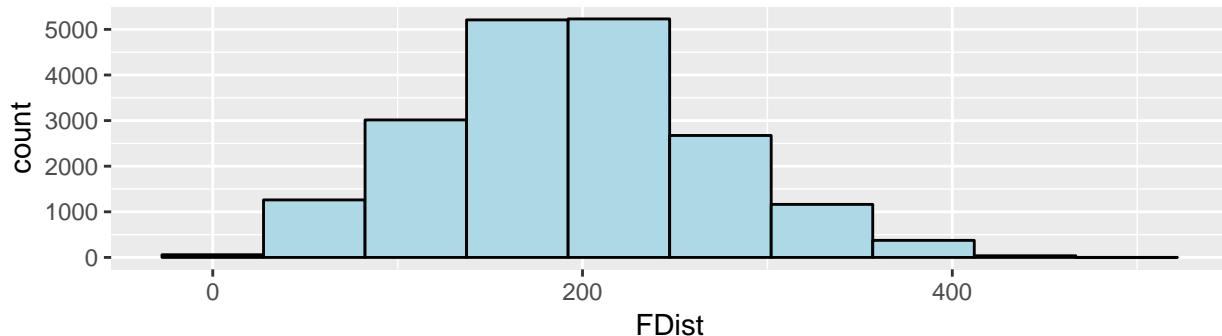
histogram



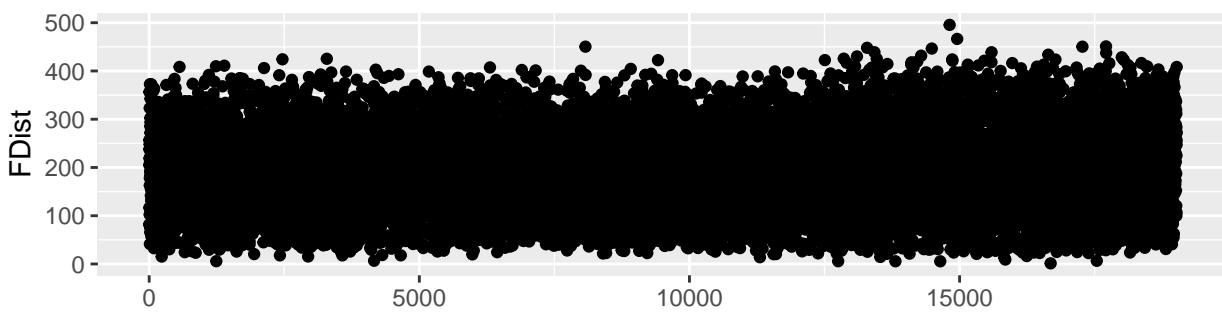
scatter



histogram



scatter



```
## [[1]]  
## TableGrob (3 x 1) "arrange": 3 grobs  
##   z   cells    name      grob
```

```

## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.723]
##
## [[2]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.814]
##
## [[3]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.905]
##
## [[4]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.996]
##
## [[5]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1087]
##
## [[6]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1178]
##
## [[7]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1269]
##
## [[8]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name           grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1360]
##
## [[9]]

```

```

## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name      grob
## 1 1 (2-2,1-1) arrange    gtable[layout]
## 2 2 (3-3,1-1) arrange    gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1451]
##
## [[10]]
## TableGrob (3 x 1) "arrange": 3 grobs
##   z   cells   name      grob
## 1 1 (2-2,1-1) arrange    gtable[layout]
## 2 2 (3-3,1-1) arrange    gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.1542]

```

Según lo que se puede ver, las variables aparentemente no tienen outliers. Utilizaremos los test para intentar identificar outliers. Primero comenzaremos comprobando si existe al menos un outlier en cada una de las variables, para ello utilizaremos el test de Grubbs, este test se considera significativo si los valores que obtenemos son menores que 0.01.

```

test.de.Grubbs = apply(mydata.numeric, 2, grubbs.test, two.sided=TRUE)
test.de.Grubbs

```

```

## $FLength
##
## Results of Hypothesis Test
## -----
##
## Alternative Hypothesis:      highest value 334.177 is an outlier
##
## Test Name:                  Grubbs test for one outlier
##
## Data:                      newX[, i]
##
## Test Statistics:            G.18696 = 6.6311297
##                            U       = 0.9976879
##
## P-value:                   6.177066e-07
##
##
## $FWidth
##
## Results of Hypothesis Test
## -----
##
## Alternative Hypothesis:      highest value 256.382 is an outlier
##
## Test Name:                  Grubbs test for one outlier
##
## Data:                      newX[, i]
##
## Test Statistics:            G.17718 = 12.7657427
##                            U       = 0.9914311
##
## P-value:                   0
##
## 
```

```

## $FSize
##
## Results of Hypothesis Test
## -----
##
## Alternative Hypothesis: highest value 5.3233 is an outlier
##
## Test Name: Grubbs test for one outlier
##
## Data: newX[, i]
##
## Test Statistics: G.17959 = 5.2862678
## U = 0.9985306
##
## P-value: 0.00235021
##
##
## $FConc
##
## Results of Hypothesis Test
## -----
##
## Alternative Hypothesis: highest value 0.893 is an outlier
##
## Test Name: Grubbs test for one outlier
##
## Data: newX[, i]
##
## Test Statistics: G.18600 = 2.8043548
## U = 0.9995865
##
## P-value: 0
##
##
## $FConc1
##
## Results of Hypothesis Test
## -----
##
## Alternative Hypothesis: highest value 0.6752 is an outlier
##
## Test Name: Grubbs test for one outlier
##
## Data: newX[, i]
##
## Test Statistics: G.5267 = 4.1674015
## U = 0.9990868
##
## P-value: 0.5836892
##
##
## $FAsym
##
## Results of Hypothesis Test

```

```

## -----
## 
## Alternative Hypothesis: highest value 575.2407 is an outlier
## 
## Test Name: Grubbs test for one outlier
## 
## Data: newX[, i]
## 
## Test Statistics: G.17122 = 9.7890727
## U      = 0.9949613
## 
## P-value: 0
## 
## 
## $FM3Long
## 
## Results of Hypothesis Test
## -----
## 
## Alternative Hypothesis: lowest value -331.78 is an outlier
## 
## Test Name: Grubbs test for one outlier
## 
## Data: newX[, i]
## 
## Test Statistics: G.14223 = 6.712250
## U      = 0.997631
## 
## P-value: 3.549083e-07
## 
## 
## $FM3Trans
## 
## Results of Hypothesis Test
## -----
## 
## Alternative Hypothesis: lowest value -205.8947 is an outlier
## 
## Test Name: Grubbs test for one outlier
## 
## Data: newX[, i]
## 
## Test Statistics: G.17755 = 9.8977328
## U      = 0.9948488
## 
## P-value: 0
## 
## 
## $FAlpha
## 
## Results of Hypothesis Test
## -----
## 
## Alternative Hypothesis: highest value 90 is an outlier

```

```

##                                     Grubbs test for one outlier
##                                     newX[, i]
##                                     G.14709 = 2.388722
##                                     U      = 0.999700
##                                     0
##                                     $FDist
##                                     Results of Hypothesis Test
## -----
##                                     Alternative Hypothesis: highest value 495.561 is an outlier
##                                     Test Name:          Grubbs test for one outlier
##                                     Data:              newX[, i]
##                                     Test Statistics:   G.14816 = 4.0376791
##                                     U      = 0.9991428
##                                     P-value:          0.9768002

```

Según los resultados del test de Grubbs, las variables FLength, FSize, FWidth, FConc, FAsym, FM3Long, FM3Trans, FAlpha. Ahora guardaremos las posiciones de los outliers y su valor.

```

aux = mydata.numeric[,c("FLength", "FWidth", "FConc", "FAsym", "FM3Long", "FM3Trans", "FAlpha")]

#Obtenemos la posicion para los valores mayores.
indices.outliers = apply(abs(aux), 2, order, decreasing=TRUE)
indices.outliers = indices.outliers[1,]
valores.outliers = mydata.numeric[indices.outliers, c("FLength", "FWidth", "FConc", "FAsym", "FM3Long", "FM3Trans", "FAlpha")]
valores.outliers = diag(as.matrix(valores.outliers))
names(valores.outliers) = c("FLength", "FWidth", "FConc", "FAsym", "FM3Long", "FM3Trans", "FAlpha")

rm(aux)

cat("Posiciones outliers:\n",
    indices.outliers, "\n",
    "Valores outliers:\n",
    valores.outliers, "\n")

## Posiciones outliers:
## 18696 17718 18600 17122 14223 17755 1714
## Valores outliers:
## 334.177 256.382 0.893 575.2407 -331.78 -205.8947 90

```

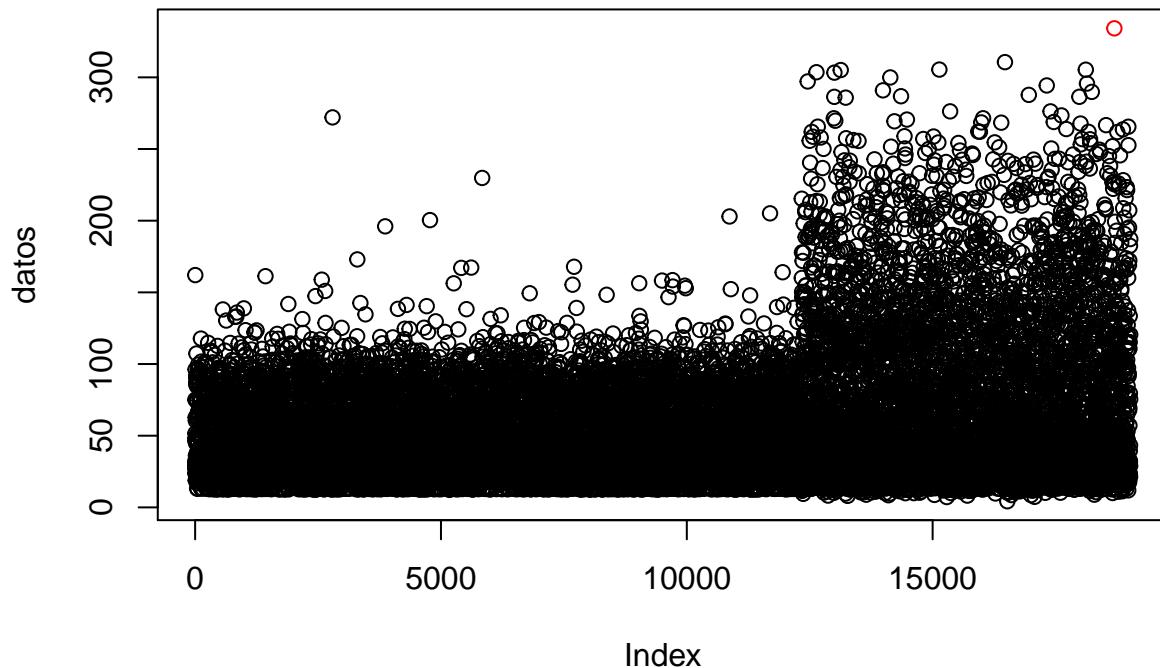
Ahora que ya tenemos los indices de los outliers, podemos mostrar un gráfico identificando los outliers de cada variable, para ello podemos utilizar la función *MiPlot_Univariate_Outliers()* que ya viene implementada.

```

MiPlot_Univariate_Outliers(mydata.numeric$FLength, indices.outliers[1], "Outlier en los datos de FLength")

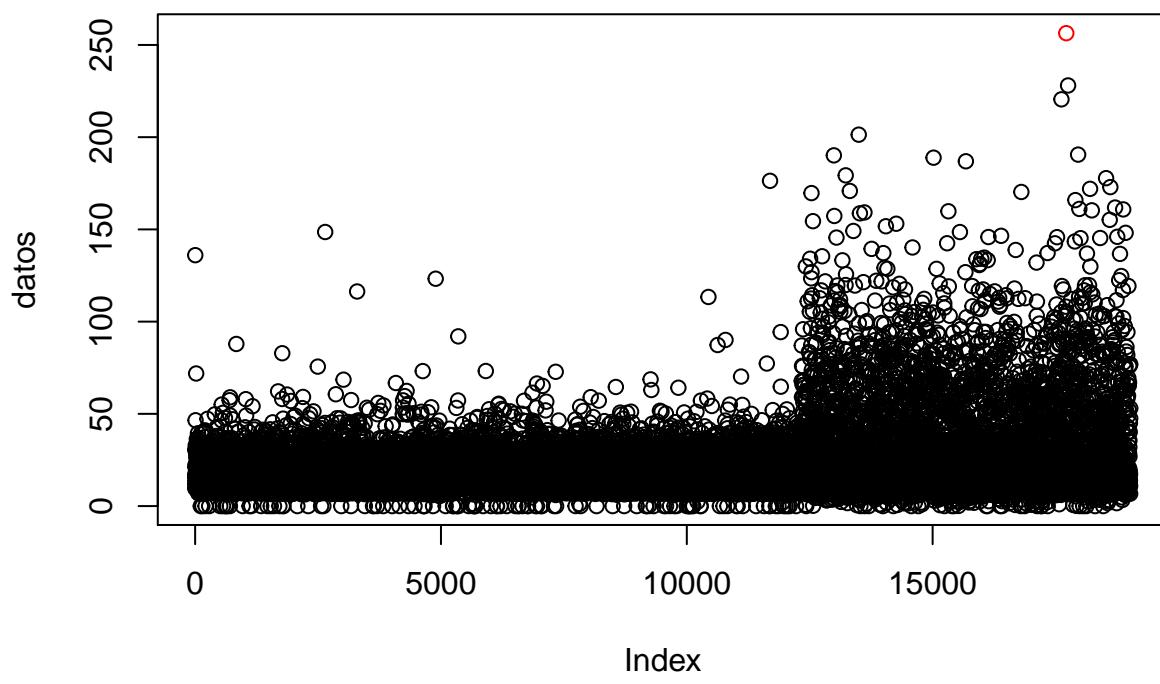
```

Outlier en los datos de FLength



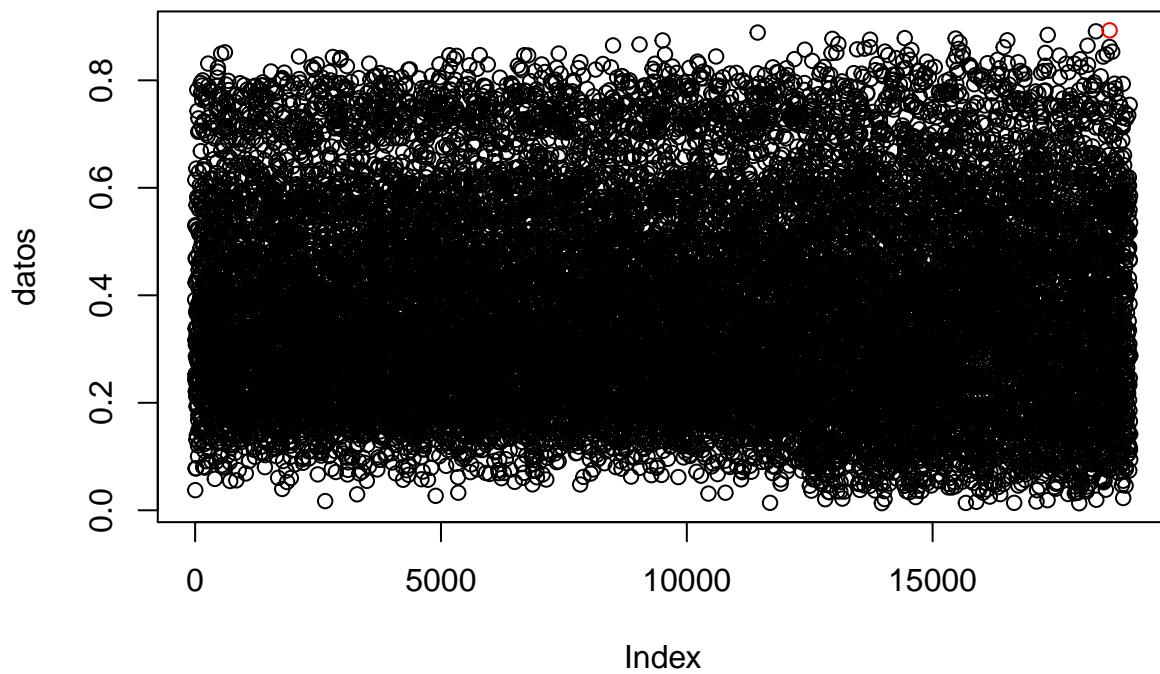
```
MiPlot_Univariate_Outliers(mydata.numeric$FWidth,indices.outliers[2],"Outlier en los datos de FWidth")
```

Outlier en los datos de FWidth



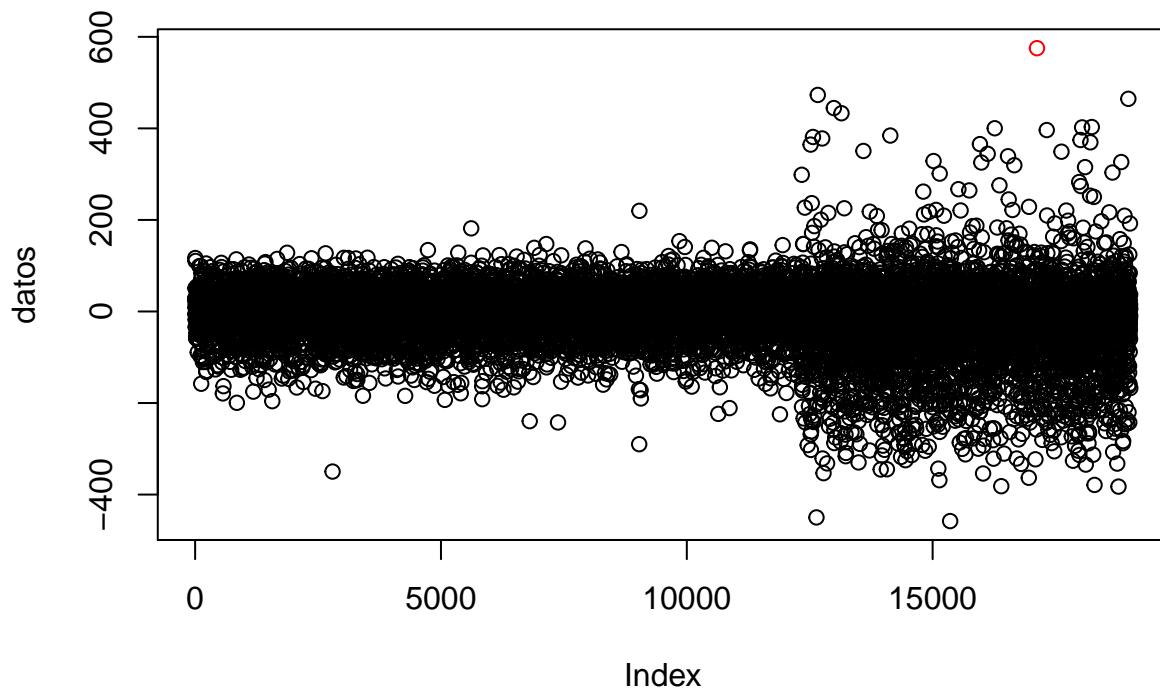
```
MiPlot_Univariate_Outliers(mydata.numeric$FConc,indices.outliers[3],"Outlier en los datos de FConc")
```

Outlier en los datos de FConc



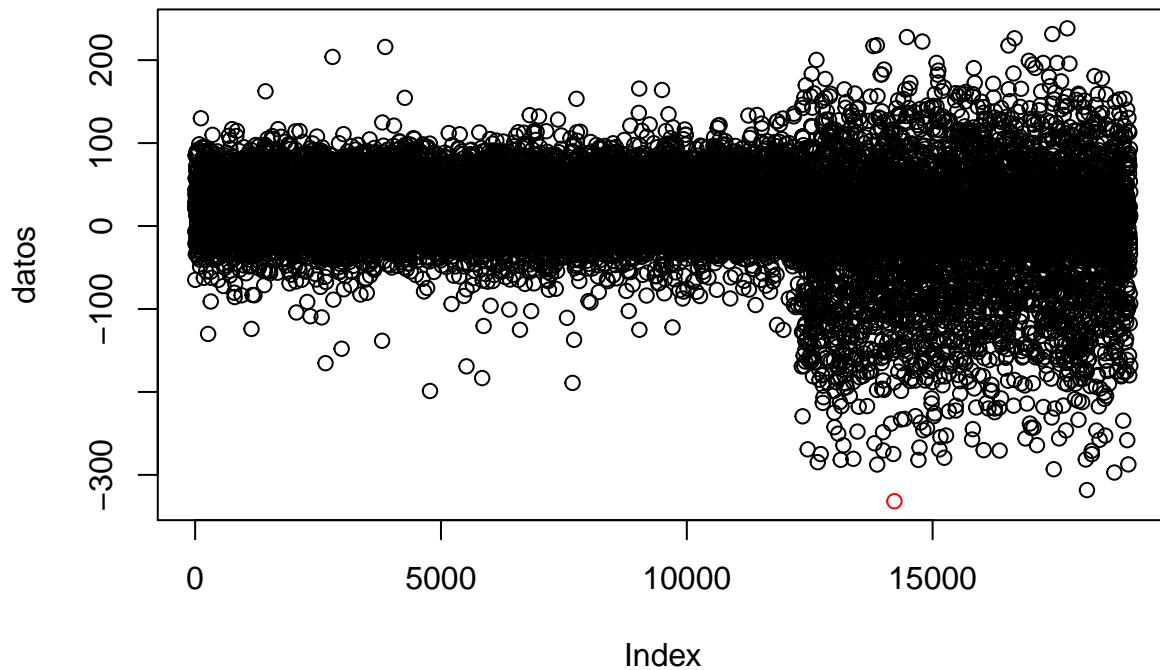
```
MiPlot_Univariate_Outliers(mydata.numeric$FAsym,indices.outliers[4],"Outlier en los datos de FAsym")
```

Outlier en los datos de FAsym



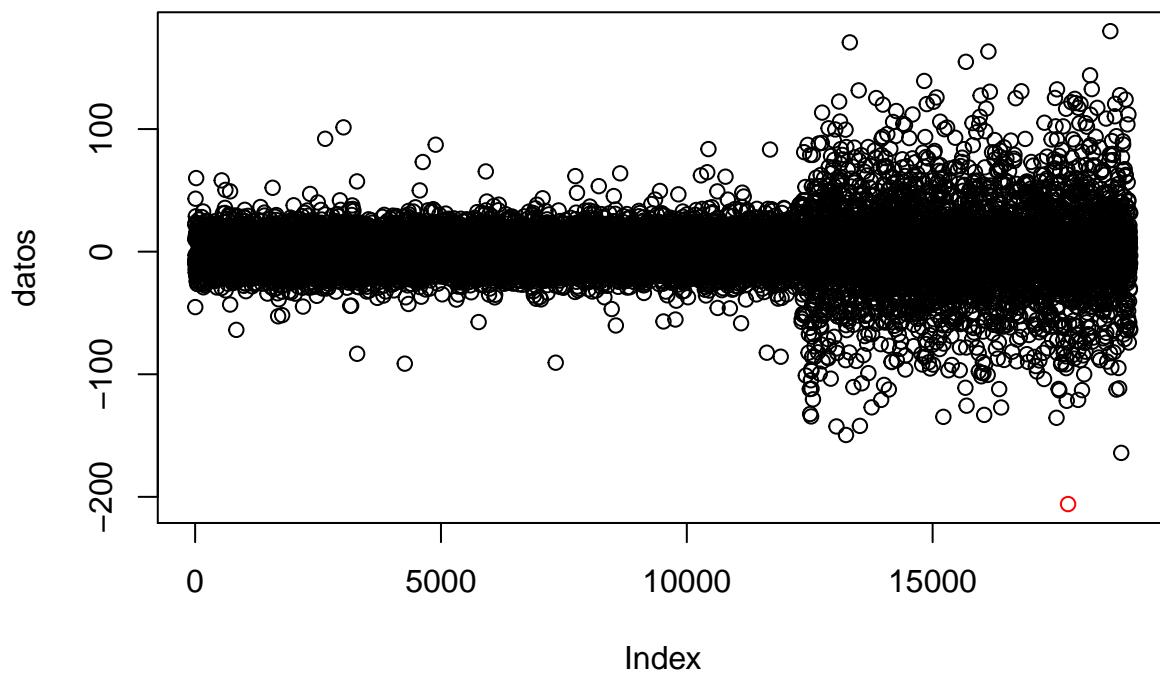
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Long,indices.outliers[5],"Outlier en los datos de FM3Long")
```

Outlier en los datos de FM3Long



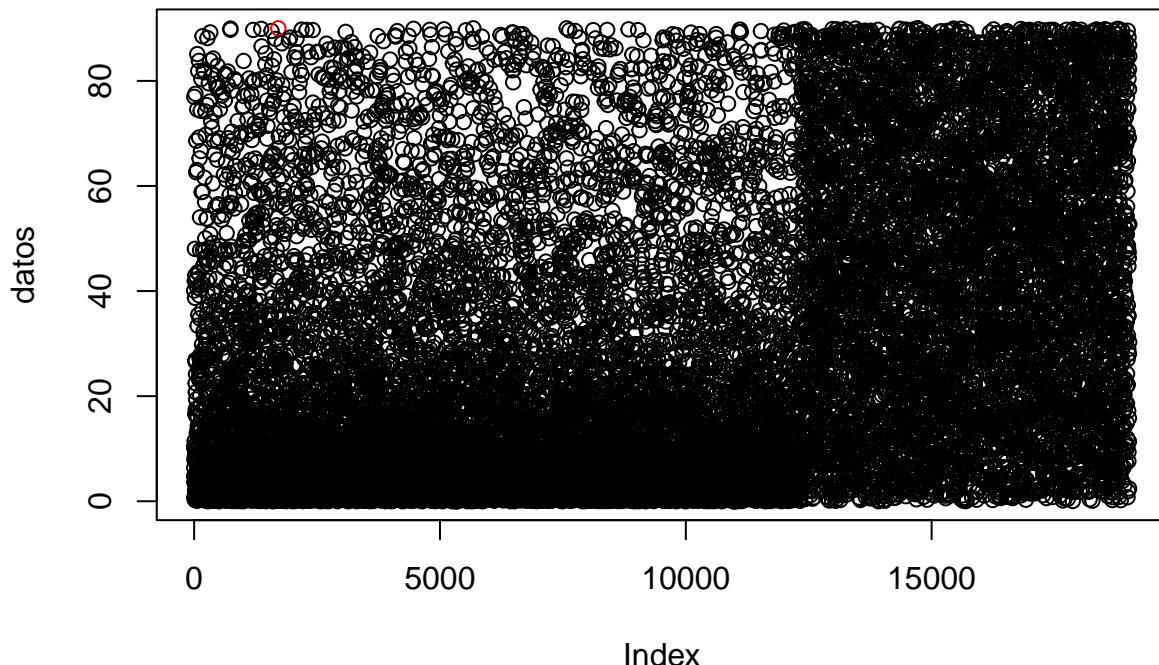
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Trans,indices.outliers[6],"Outlier en los datos de FM3Trans")
```

Outlier en los datos de FM3Trans



```
MiPlot_Univariate_Outliers(mydata.numeric$FAlpha,indices.outliers[7],"Outlier en los datos de FAlpha")
```

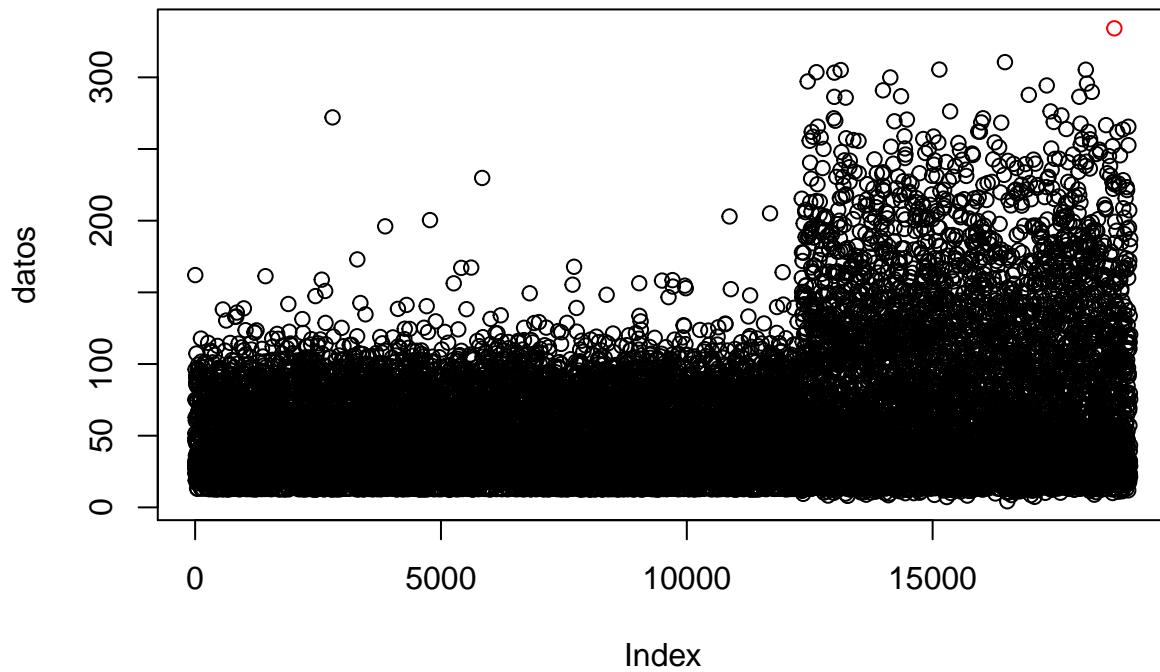
Outlier en los datos de FAlpha



Gracias a las gráficas, podemos saber que en alguno de los datos los outliers detectados no son reales, como por ejemplo en FConc o FAlpha. Todo esto que hemos hecho anteriormente para poder mostrar los datos ya se encuentra implementado en la función *MiPlot_resultados_TestGrubbs()*. Ahora utilizaremos esta función para mostrar los resultados del test de Grubbs con todas las funciones, ya que es posible que en alguna de las variables que hemos rechazado se haya producido el error de masking, al igual que en algunas de las variables sí se han identificado outliers en lugares donde no los había.

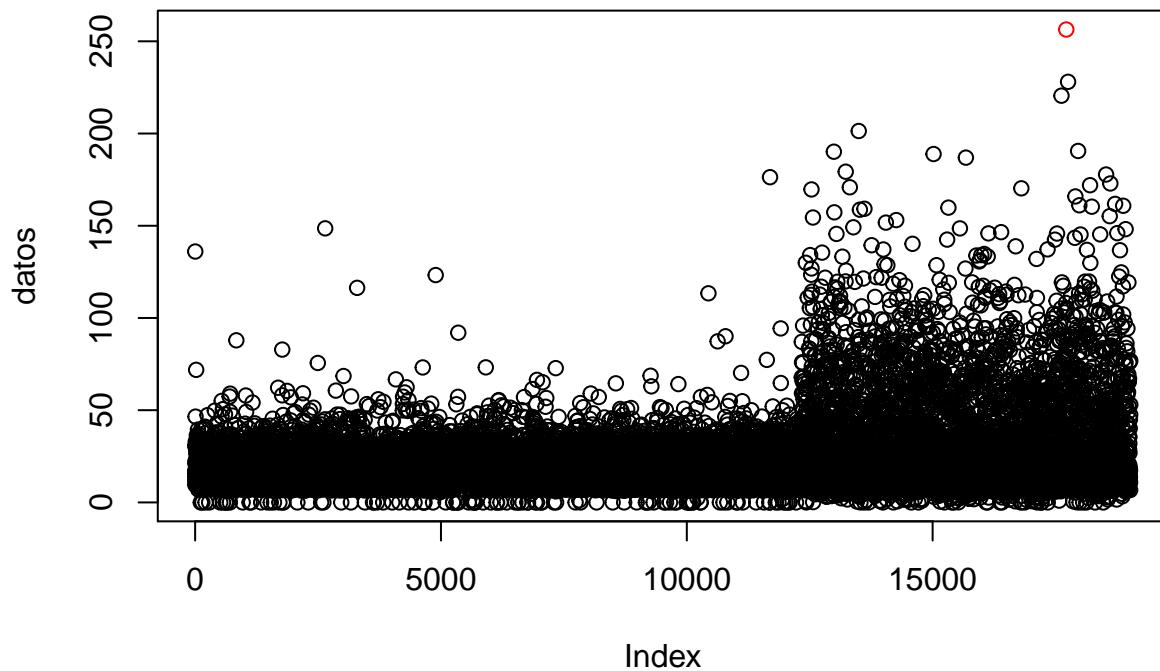
```
aux = mydata.numeric  
apply(aux, 2, MiPlot_resultados_TestGrubbs)  
  
## p.value: 6.177066e-07  
## ?ndice de outlier: 18696  
## Valor del outlier: 334.177
```

Test de Grubbs



```
## p.value: 0  
## ?ndice de outlier: 17718  
## Valor del outlier: 256.382
```

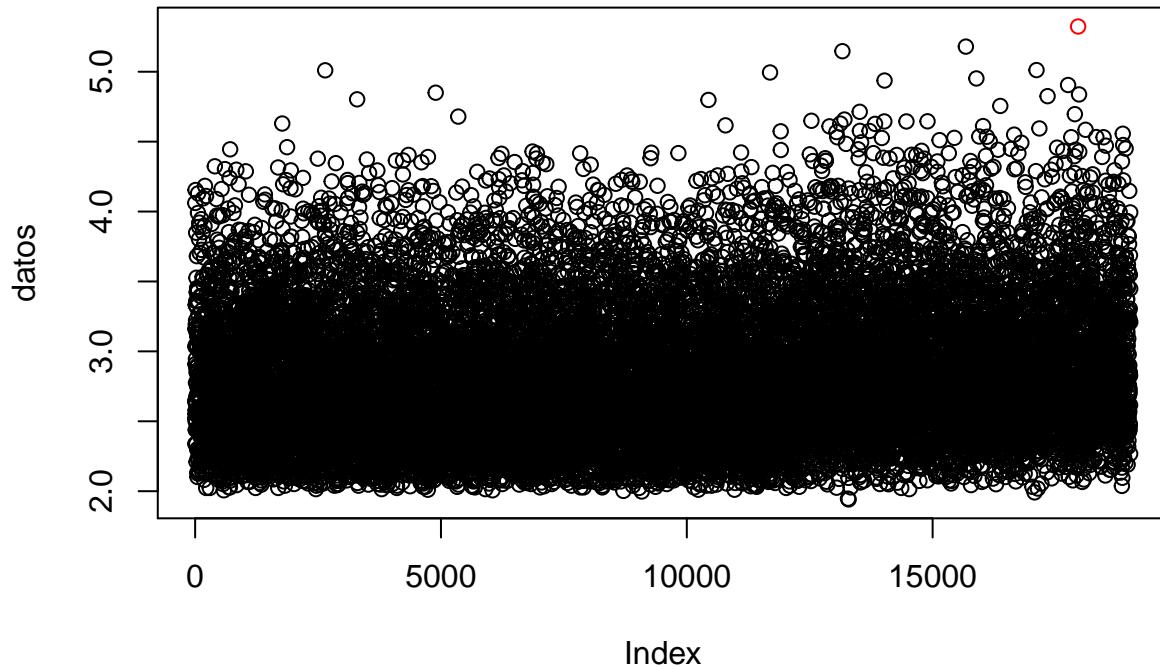
Test de Grubbs



```
## p.value: 0.00235021  
## ?ndice de outlier: 17959
```

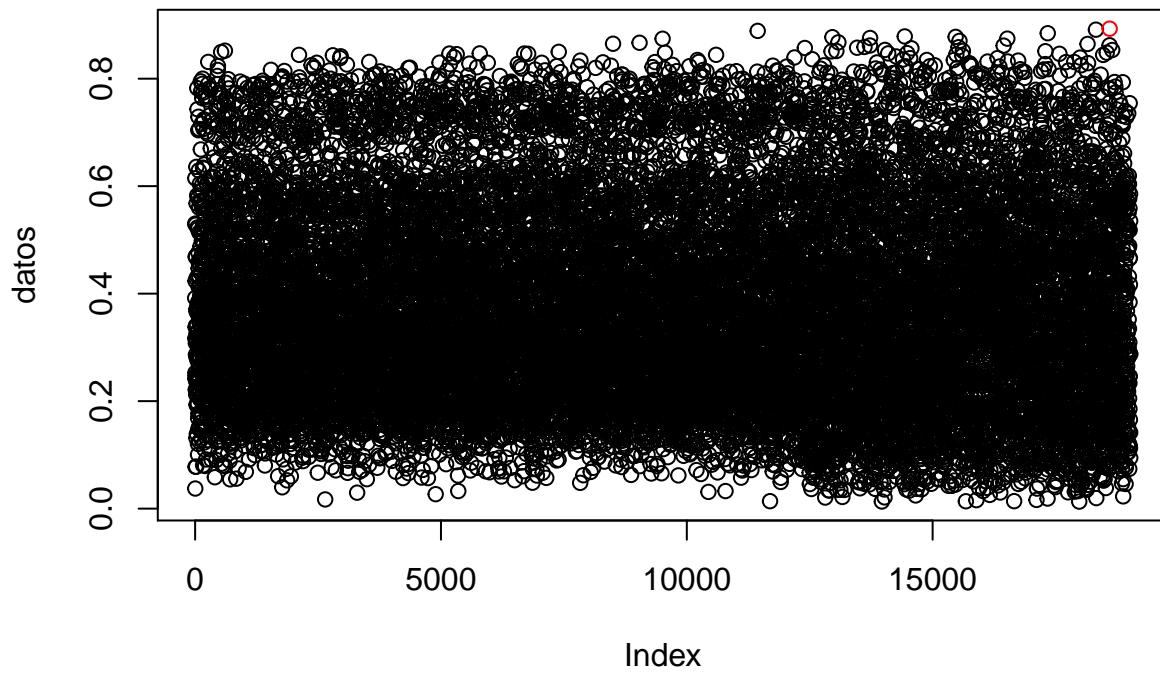
```
## Valor del outlier: 5.3233
```

Test de Grubbs



```
## p.value: 0  
## ?ndice de outlier: 18600  
## Valor del outlier: 0.893
```

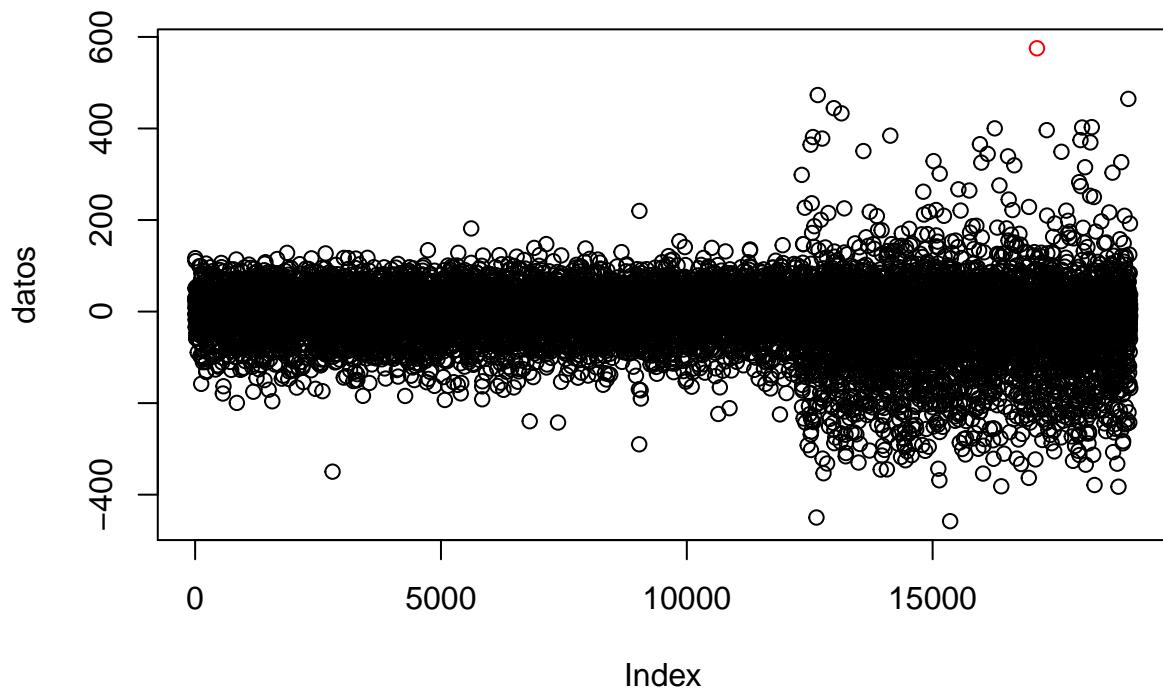
Test de Grubbs



```
## p.value: 0.5836892
```

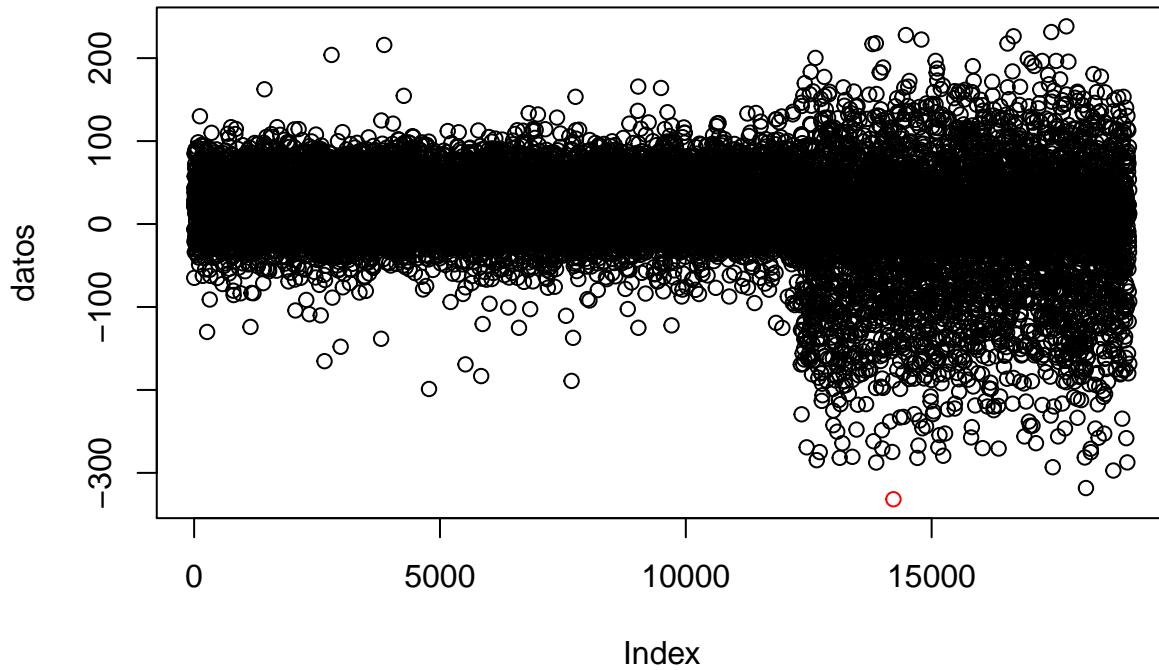
```
## No hay outliers p.value: 0  
## ?ndice de outlier: 17122  
## Valor del outlier: 575.2407
```

Test de Grubbs



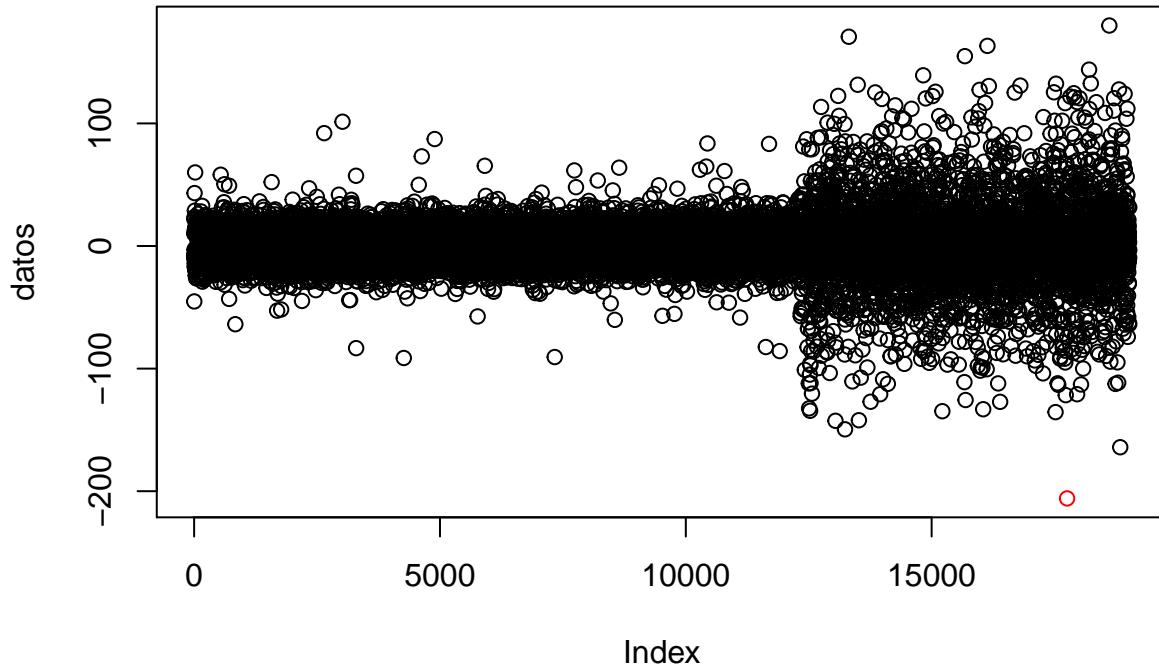
```
## p.value: 3.549083e-07  
## ?ndice de outlier: 14223  
## Valor del outlier: -331.78
```

Test de Grubbs



```
## p.value: 0  
## ?ndice de outlier: 17755  
## Valor del outlier: -205.8947
```

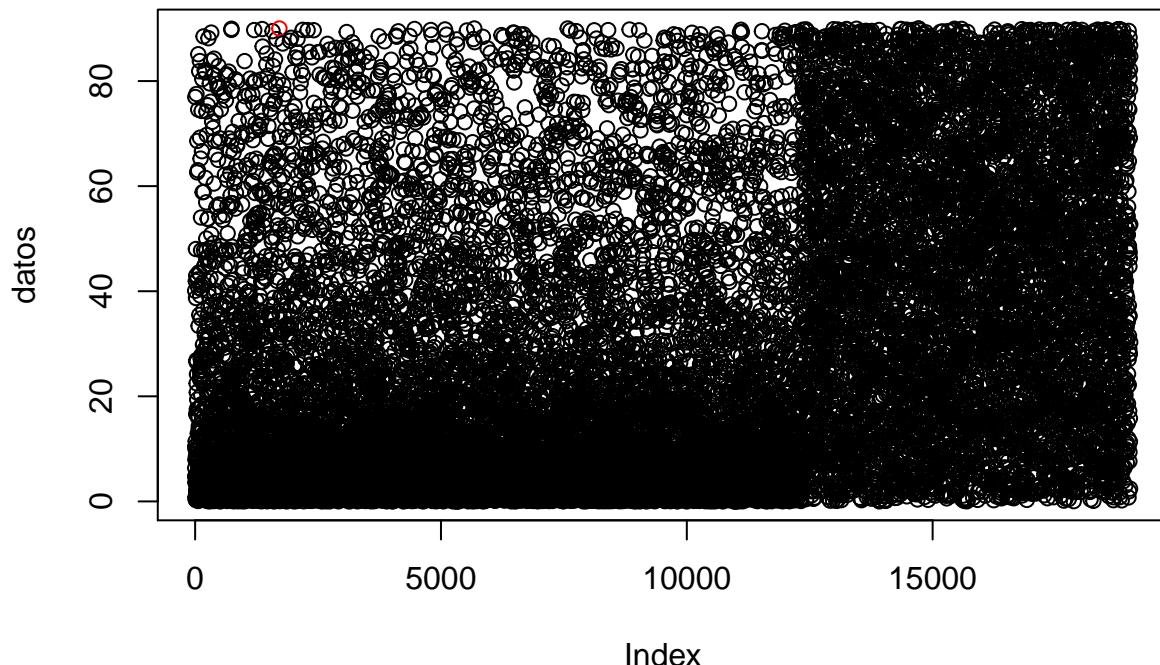
Test de Grubbs



```
## p.value: 0  
## ?ndice de outlier: 1714
```

```
## Valor del outlier: 90
```

Test de Grubbs



```
## p.value: 0.9768002
## No hay outliers
## NULL
rm(aux)
```

Las variables que no se han pintado son aquellas que según el test de Grubbs no contienen ningún outlier. Lo siguiente que vamos a hacer es utilizar el test de Rosner para ver si hay un número de outliers que k, el cual definimos nosotros. Para ello utilizaremos la función *rosnerTest()*.

```
# Función para obtener los valores interesantes.
obtener_valores_Rosner = function(data,k=4){
  rTest = rosnerTest(data,k=k)
  bool.outlier = rTest$all.stats$Outlier
  indices.outliers.rosner = rTest$all.stats$Obs.Num
  resultados = list(bools=bool.outlier,indices=indices.outliers.rosner)
  resultados
}

test.de.Rosner = apply(mydata.numeric,2,obtener_valores_Rosner)
test.de.Rosner

## $FLength
## $FLength$bools
## [1] TRUE TRUE TRUE TRUE
##
## $FLength$indices
## [1] 18696 16471 15137 18115
##
##
```

```

## $FWidth
## $FWidth$bools
## [1] TRUE TRUE TRUE TRUE
##
## $FWidth$indices
## [1] 17718 17755 17620 13497
##
##
## $FSize
## $FSize$bools
## [1] TRUE TRUE TRUE FALSE
##
## $FSize$indices
## [1] 17959 15676 13165 17113
##
##
## $FConc
## $FConc$bools
## [1] FALSE FALSE FALSE FALSE
##
## $FConc$indices
## [1] 18600 18321 11442 17341
##
##
## $FConc1
## $FConc1$bools
## [1] FALSE FALSE FALSE FALSE
##
## $FConc1$indices
## [1] 5267 5172 15482 13083
##
##
## $FAsym
## $FAsym$bools
## [1] TRUE TRUE TRUE TRUE
##
## $FAsym$indices
## [1] 17122 12664 18980 15356
##
##
## $FM3Long
## $FM3Long$bools
## [1] TRUE TRUE TRUE TRUE
##
## $FM3Long$indices
## [1] 14223 18139 18696 17463
##
##
## $FM3Trans
## $FM3Trans$bools
## [1] TRUE TRUE TRUE TRUE
##
## $FM3Trans$indices
## [1] 17755 18614 13316 18836

```

```

## 
## 
## $FAlpha
## $FAlpha$bools
## [1] FALSE FALSE FALSE FALSE
##
## $FAlpha$indices
## [1] 1714 5678 7577 14709
##
##
## $FDist
## $FDist$bools
## [1] FALSE FALSE FALSE FALSE
##
## $FDist$indices
## [1] 14816 14956 17711 8075

```

Con este test, podemos ver que para algunas de las variables no obtenemos ningún outlier, como por ejemplo FConc, para la cual antes sí obteníamos que había un outlier. Ahora utilizaremos la función *MiPlot_Univariate_Outliers()* al igual que antes para representar los datos de aquellos que sí que haya encontrado outliers, aunque solo con los datos que sí sean realmente outliers.

```

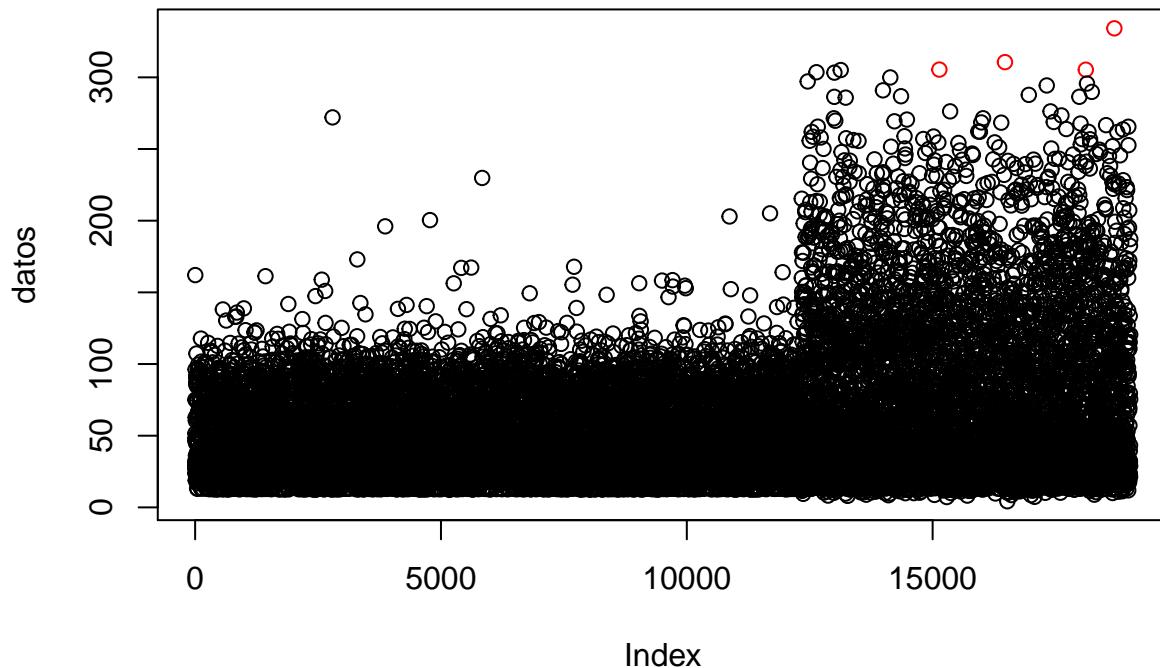
# Función para devolver la posición de los outliers reales (TRUE) del test de Rosner
obtener_outlier_reales_Rosner = function(data=list()){
  m_data = data.frame(bool=data$bools,ind=data$indices)
  resultados = as.vector(subset(m_data,bool==TRUE,select=ind)$ind)
  resultados
}

indices.rosner = lapply(test.de.Rosner, obtener_outlier_reales_Rosner)
# Eliminamos los que no tienen resultados.
indices.rosner$FConc = NULL
indices.rosner$FConc1 = NULL
indices.rosner$FAlpha = NULL
indices.rosner$FDist = NULL

# Dibujamos los resultados.
MiPlot_Univariate_Outliers(mydata.numeric$FLength, indices.rosner$FLength, "Outliers Rosner FLength")

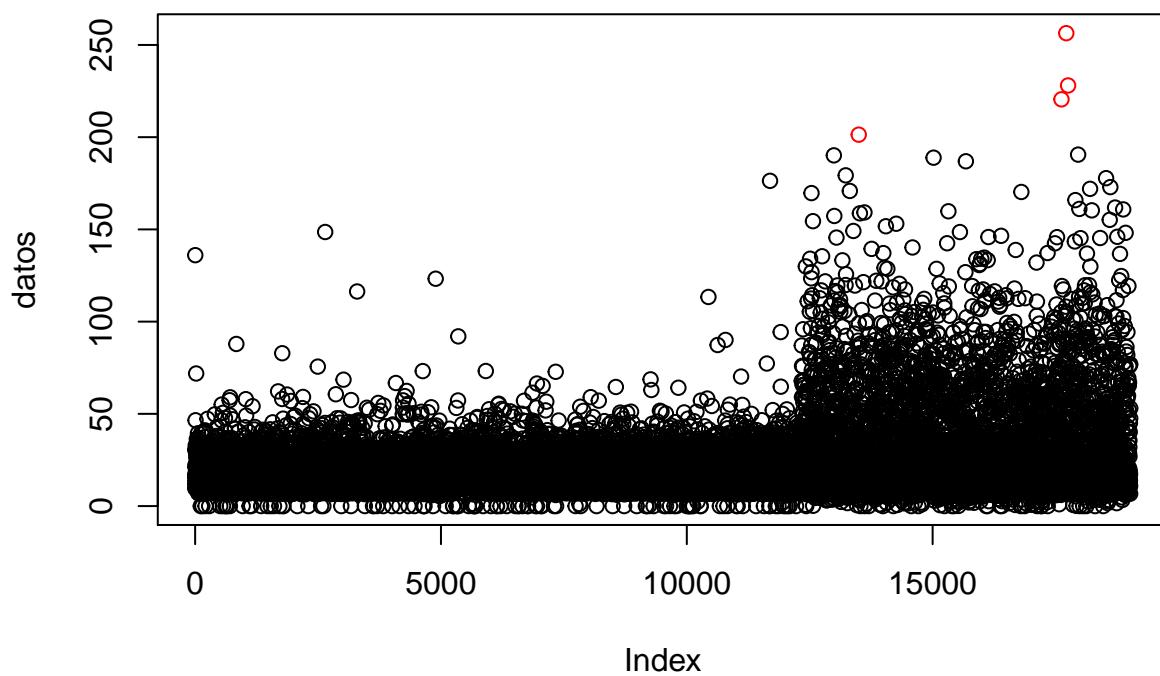
```

Outliers Rosner FLength



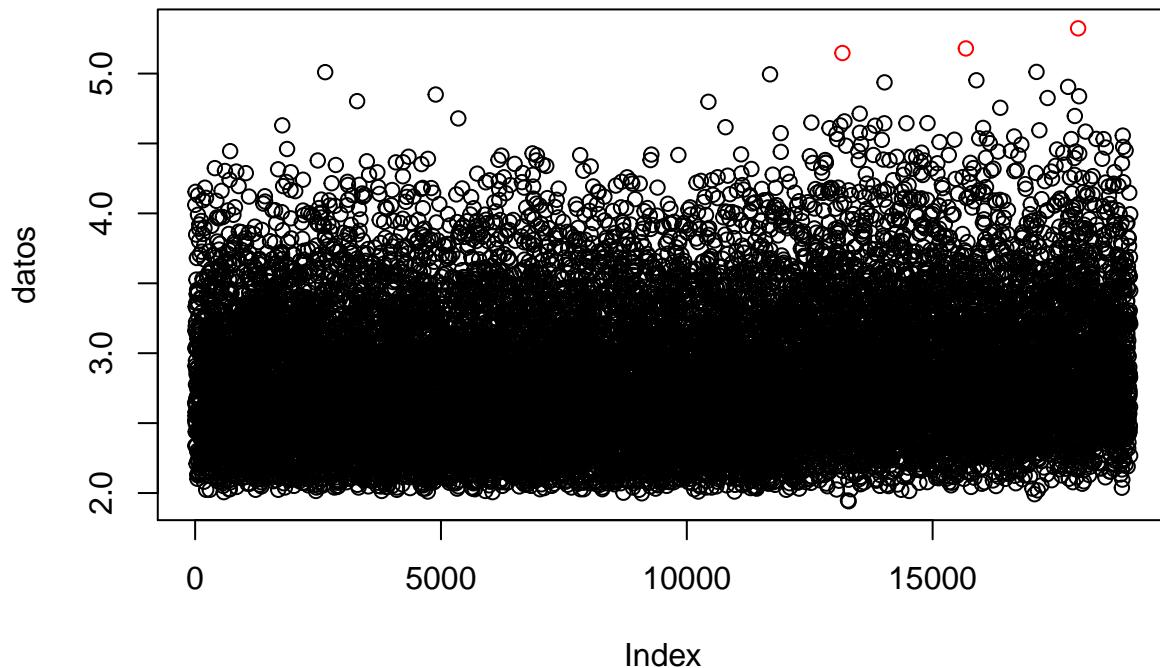
```
MiPlot_Univariate_Outliers(mydata.numeric$FWidth, indices.rosner$FWidth, "Outliers Rosner FWidth")
```

Outliers Rosner FWidth



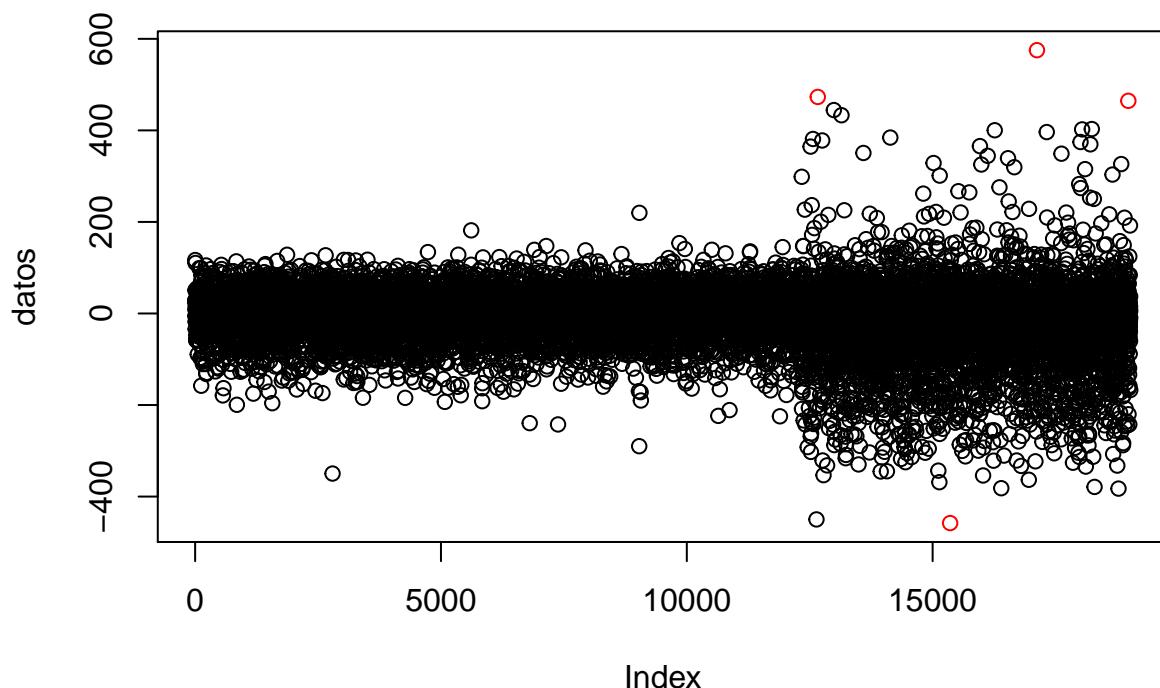
```
MiPlot_Univariate_Outliers(mydata.numeric$FSize, indices.rosner$FSize, "Outliers Rosner FSize")
```

Outliers Rosner FSize



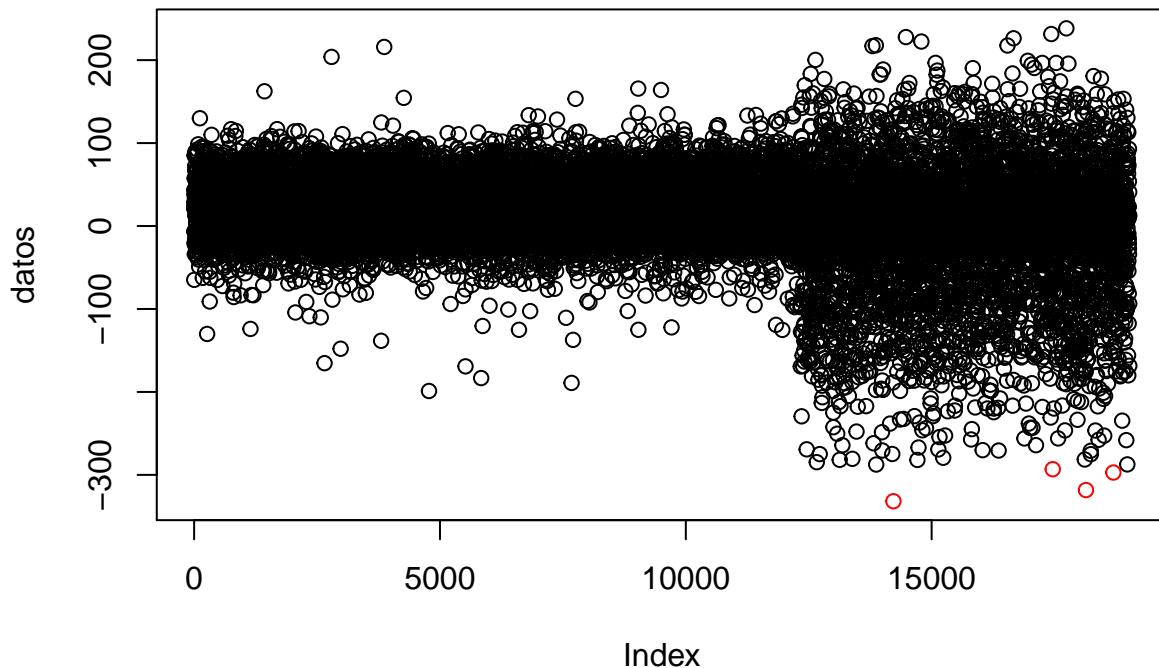
```
MiPlot_Univariate_Outliers(mydata.numeric$FAsym, indices.rosner$FAsym, "Outliers Rosner FAsym")
```

Outliers Rosner FAsym



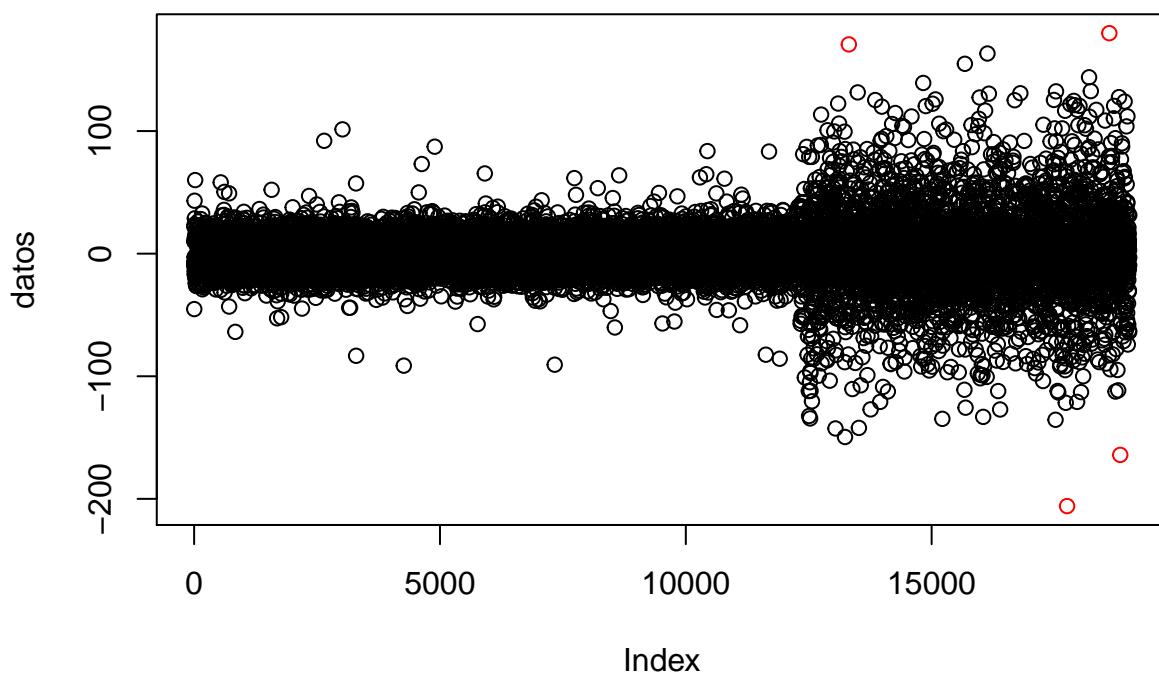
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Long, indices.rosner$FM3Long, "Outliers Rosner FM3Long")
```

Outliers Rosner FM3Long



```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Trans, indices.rosner$FM3Trans, "Outliers Rosner FM3Trans")
```

Outliers Rosner FM3Trans



Como se puede ver, el test de Rosner ofrece mejores resultados para obtener outliers en este conjunto de datos, ya que los datos considerados como outliers tienen bastante sentido. Todo este proceso realizado anteriormente se encuentra ya hecho en la función *MiPlot_resultados_TestRosner()*, veamos los resultados que obtiene con nuestro dataset.

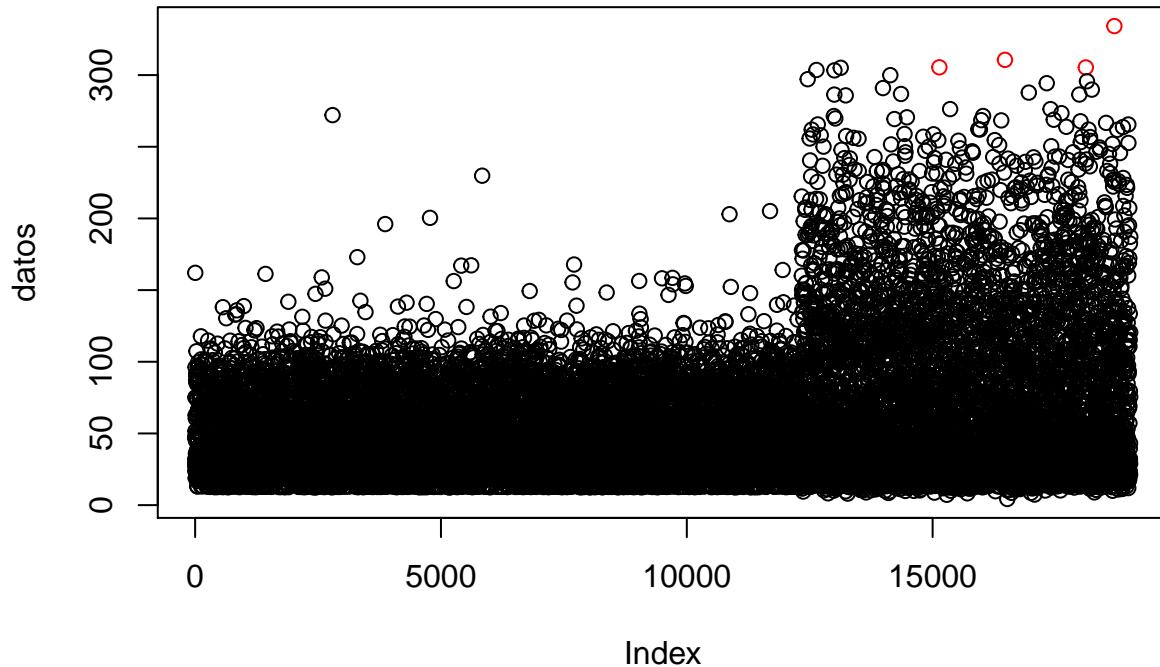
```

apply(mydata.numeric, 2, MiPlot_resultados_TestRosner)

##
## Test de Rosner
## ?ndices de las k-mayores desviaciones de la media: 18696 16471 15137 18115
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE
## Los ?ndices de los outliers son: 18696 16471 15137 18115
## Los valores de los outliers son: 334.177 310.61 305.422 305.324

```

Test de Rosner

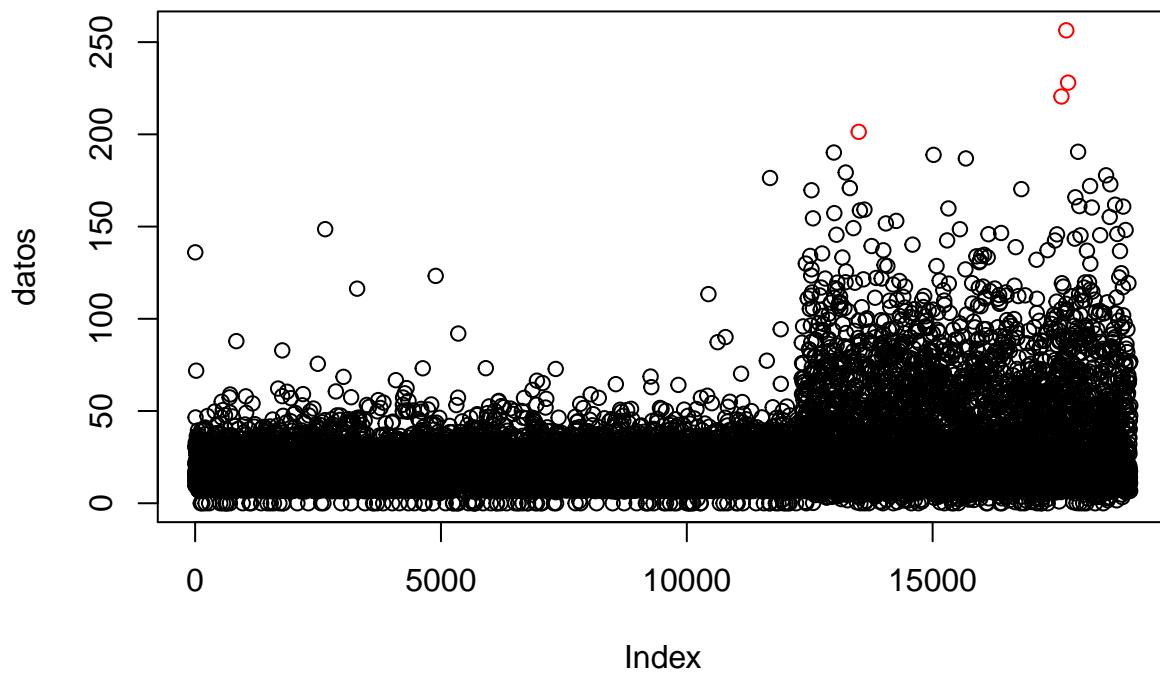


```

##
## Test de Rosner
## ?ndices de las k-mayores desviaciones de la media: 17718 17755 17620 13497
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE
## Los ?ndices de los outliers son: 17718 17755 17620 13497
## Los valores de los outliers son: 256.382 228.0385 220.5144 201.364

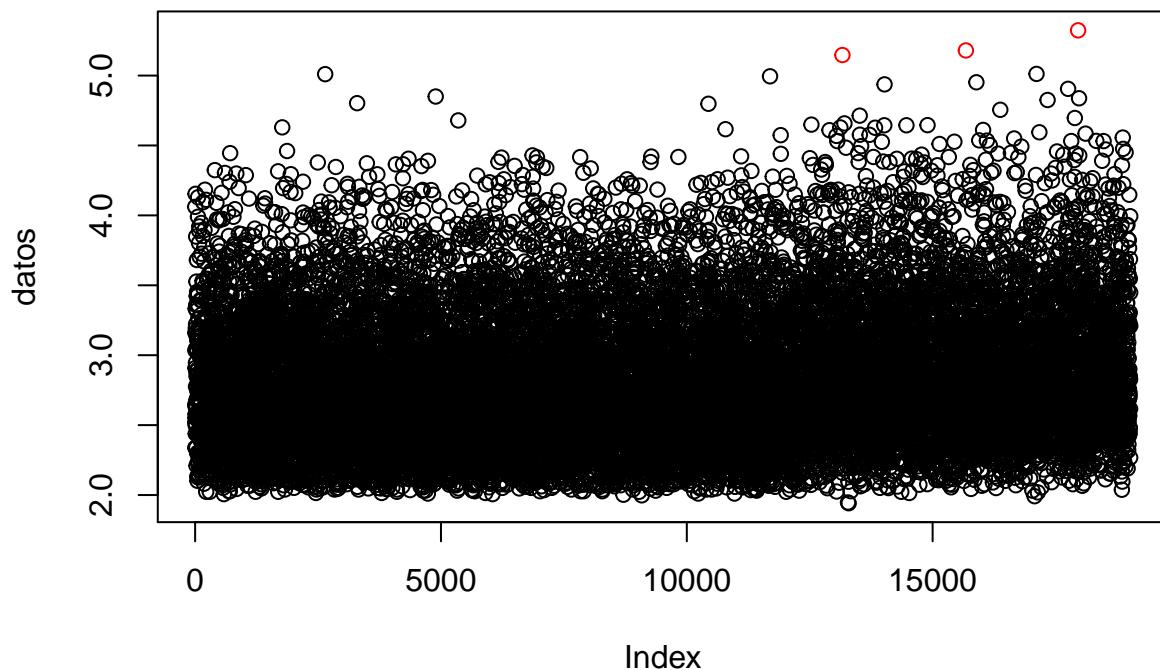
```

Test de Rosner



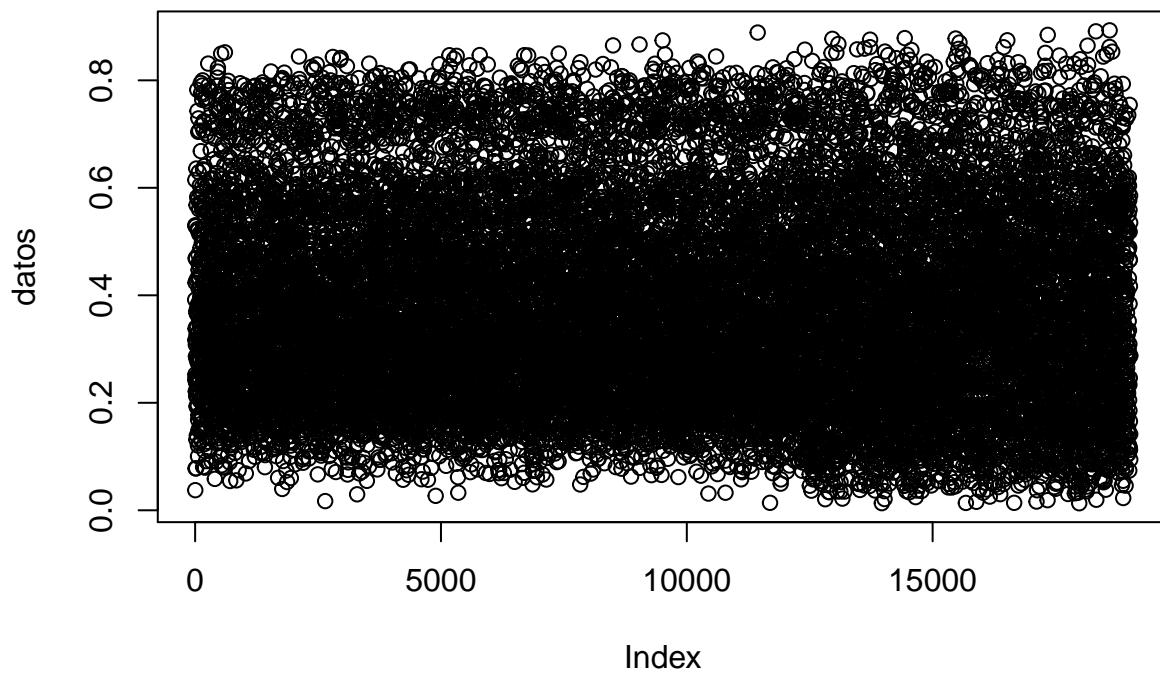
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17959 15676 13165 17113  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE FALSE  
## Los ?ndices de los outliers son: 17959 15676 13165  
## Los valores de los outliers son: 5.3233 5.1795 5.1467
```

Test de Rosner



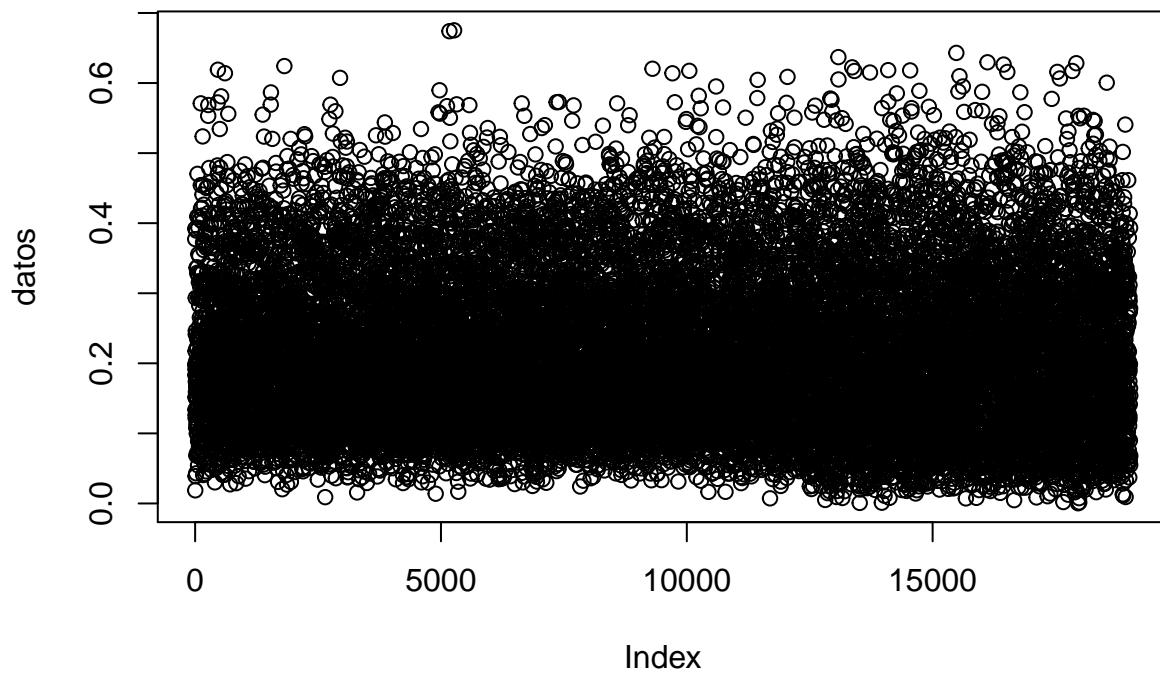
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 18600 18321 11442 17341  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



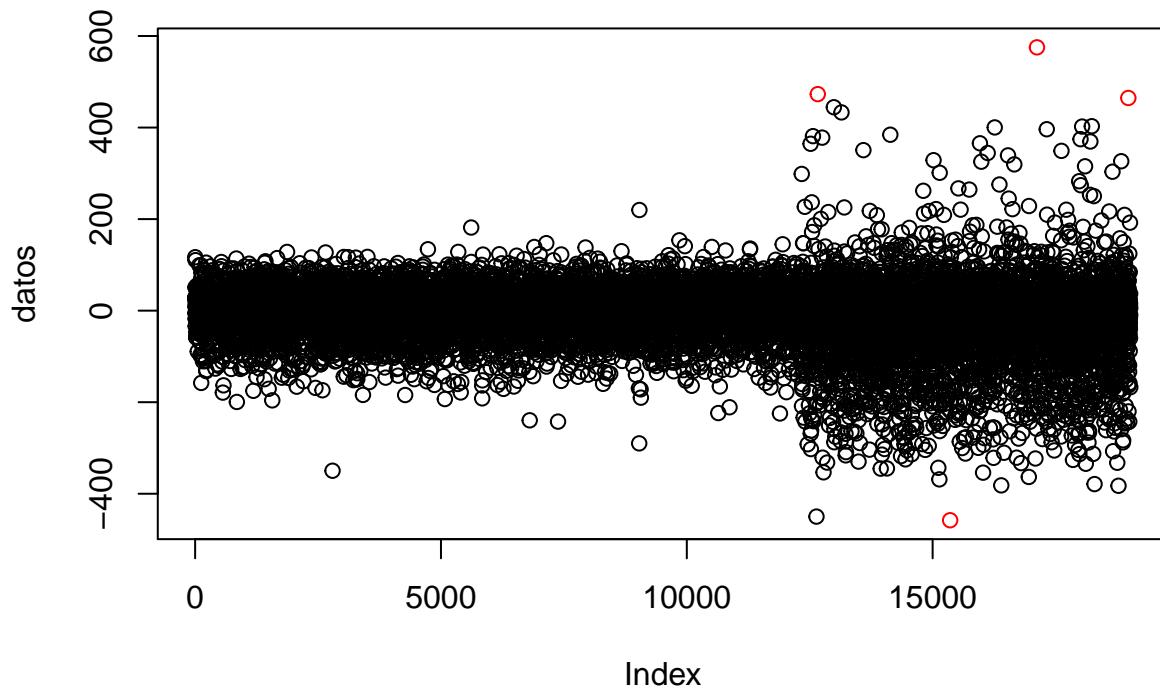
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 5267 5172 15482 13083  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



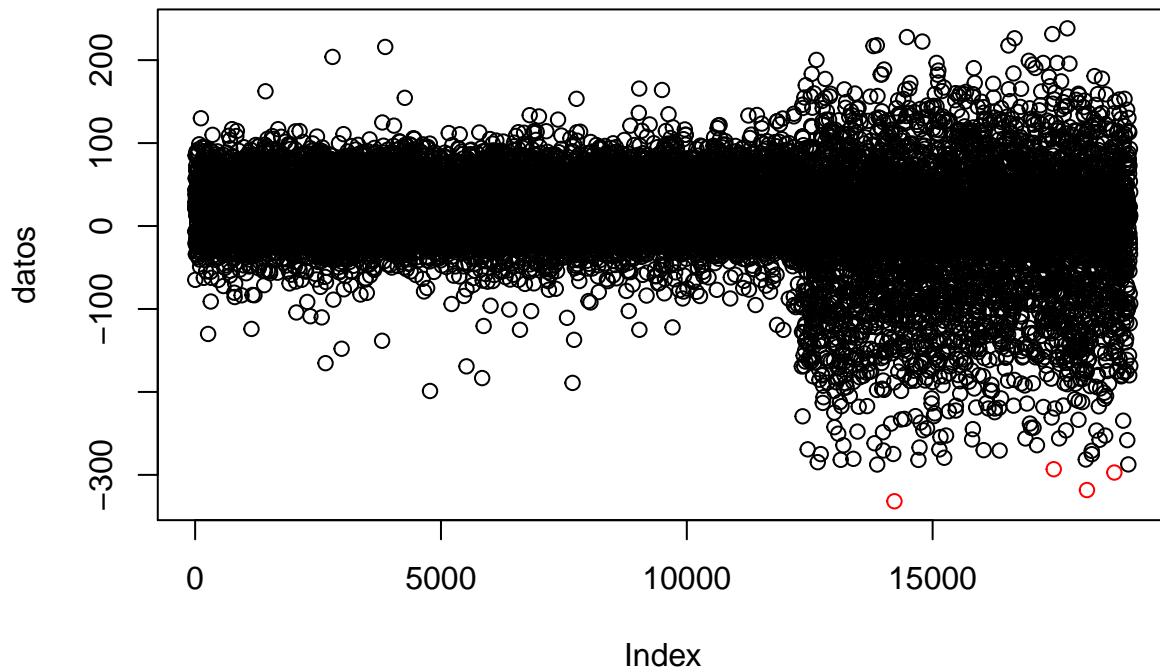
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17122 12664 18980 15356  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 17122 12664 18980 15356  
## Los valores de los outliers son: 575.2407 473.0654 464.631 -457.9161
```

Test de Rosner



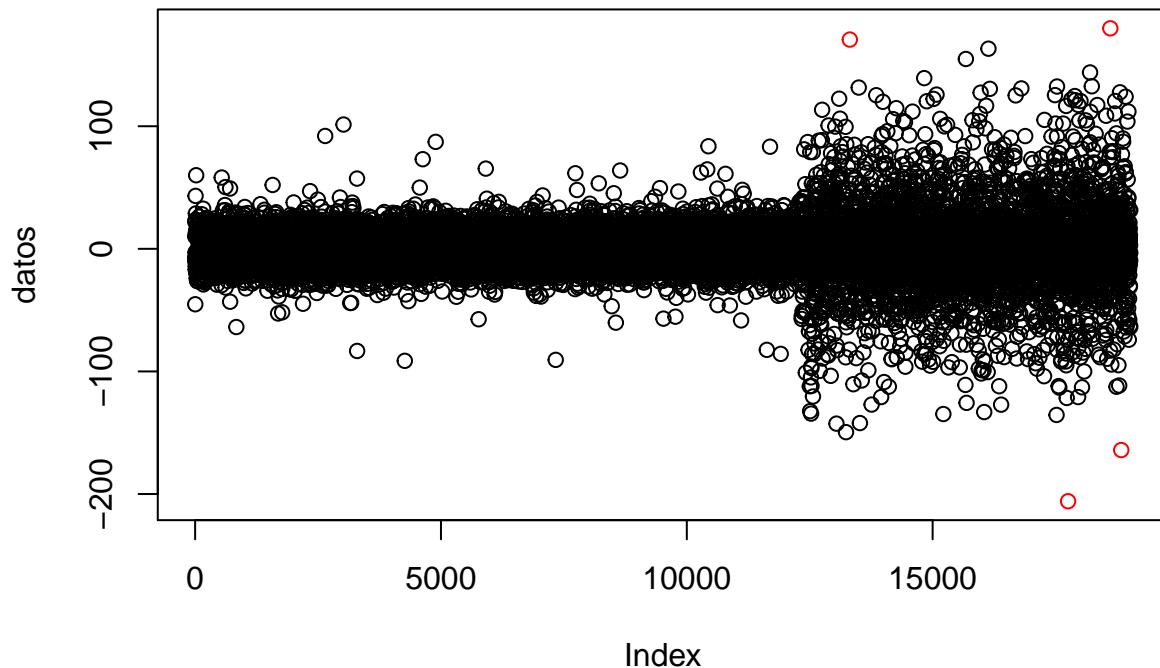
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 14223 18139 18696 17463  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 14223 18139 18696 17463  
## Los valores de los outliers son: -331.78 -318.3002 -297.1717 -293.1762
```

Test de Rosner



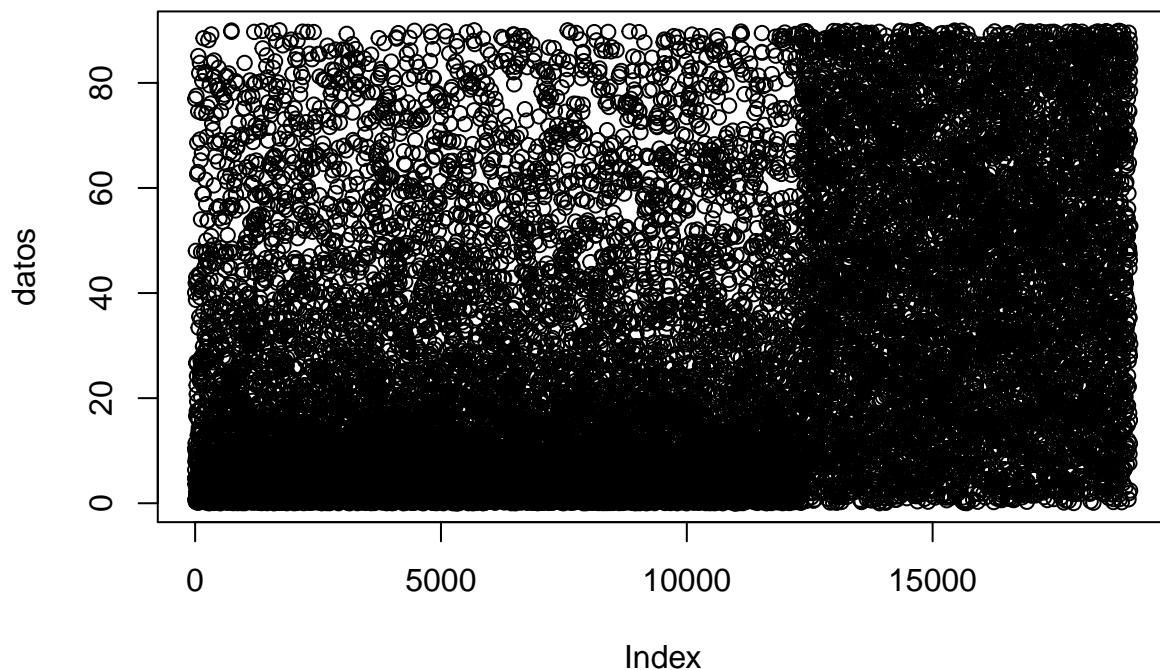
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17755 18614 13316 18836  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 17755 18614 13316 18836  
## Los valores de los outliers son: -205.8947 179.851 170.692 -164.14
```

Test de Rosner



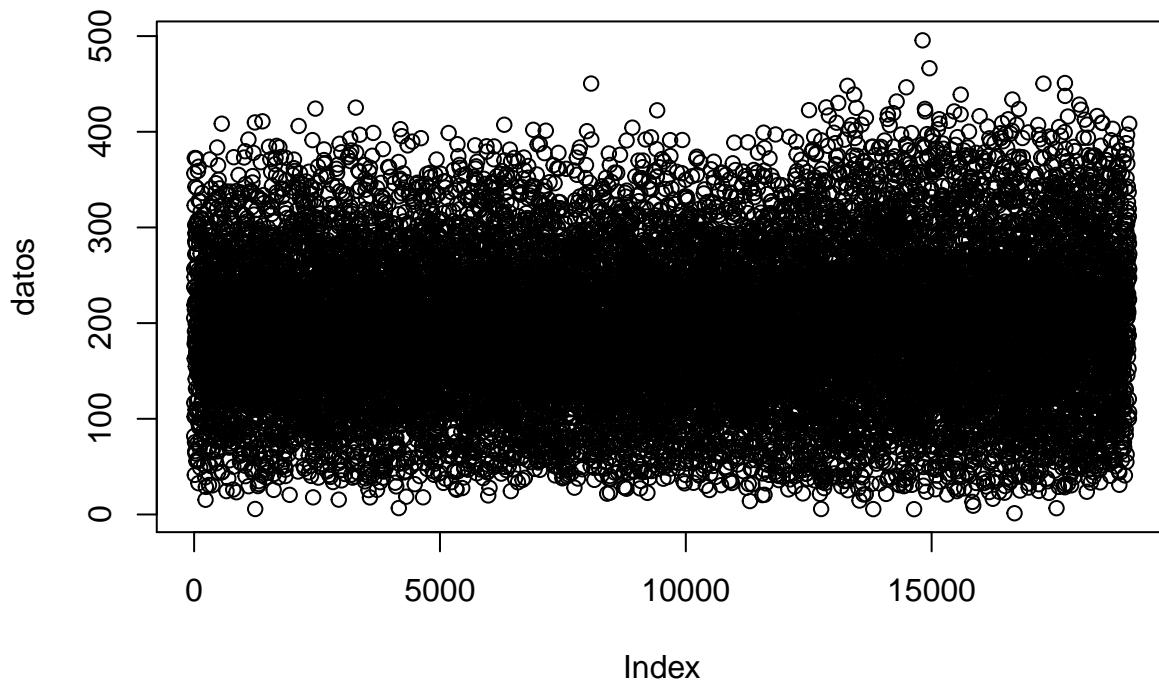
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 1714 5678 7577 14709  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 14816 14956 17711 8075  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



```
## NULL
```

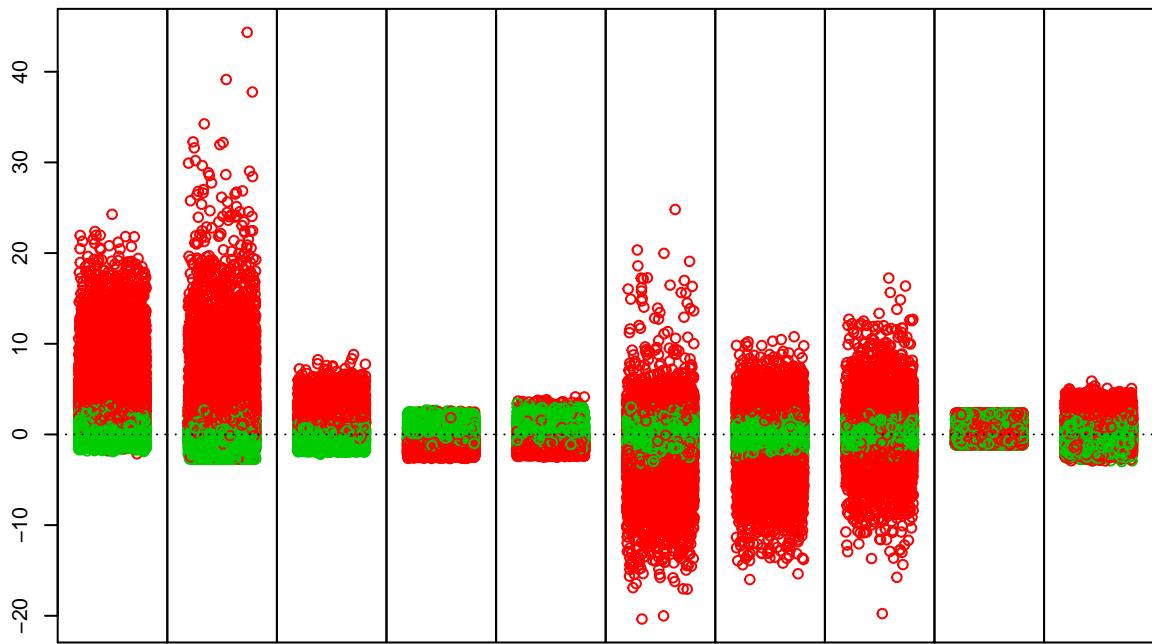
Se puede apreciar, que a diferencia de la función que utiliza el test de Grubbs, esta muestra todas las variables, y dibuja los outliers en rojo, si el test de Grubbs realmente no ha encontrado ning n outlier, ninguno de los puntos ser n rojos para esa variable.

Detecci n de outliers multivariados

En este apartado vamos a detectar en nuestro conjunto de datos outliers multivariados, es decir, que la combinaci n de los valores de varias columnas sea an malo, aunque dichos valores de las columnas no tienen porque ser an malos por si solos. Para detectar dichos outliers, utilizaremos el paquete mvoutliers que utiliza la distancia de Mahalanobis con MCD para detectar si se trata de un outlier o no. Veamos como se hace.

```
set.seed(12)
mvoulier.plot = uni.plot(mydata.numeric.scaled,symb = FALSE, alpha=0.05)
```

```
FLength FWidth FSize FConc FConc1 FAsym FM3Long M3Trans FAlpha FDist
```



```
head(mvoulier.plot$outliers, 10)
```

```
##      1      2      3      4      5      6      7      8      9      10 
## FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

Como se puede ver en el gráfico, una gran cantidad de los datos son considerados outliers según este método. Lo siguiente que vamos a hacer es guardar dichos outliers, y con ellos calcularemos el número de outliers multivariantes y el índice de estos.

```
is.MCD.outlier = mvoulier.plot$outliers
numero.de.outliers.MCD = length(which(is.MCD.outlier))
cat("número de outliers multivariante: ", numero.de.outliers.MCD, "\n")
```

```
## número de outliers multivariante: 6574
indices.MCD.outliers = which(is.MCD.outlier)
cat("índices de los primeros outliers multivariante:\n",
    head(indices.MCD.outliers, 10), "\n")
```

```
## índices de los primeros outliers multivariante:
## 3 5 9 11 15 21 22 23 25 26
```

Ahora, vamos a intentar saber cuanta es la desviación de los datos que son considerados outliers conforme a la media, para ello vamos a utilizar el dataset con los datos escalados, y miraremos solamente los índices de los datos que son outliers.

```
head(mydata.numeric.scaled[indices.MCD.outliers, ], 50)
```

```
##          FLength        FWidth        FSize        FConc        FConc1 
## 3  2.568210046  6.205695216  2.61571429 -1.87583375 -1.77319444 
## 5  0.516608545  0.476371251  0.71113838 -0.34749728 -0.28465213 
## 9  1.014580273  1.326472207  2.81207541 -1.65429607 -1.58950199 
## 11 0.224866722  0.421313096  1.07508356 -0.72657286 -0.80134372 
## 15 0.170137395  0.595503121  0.78159986  0.47902971  0.28723768 
## 21 0.906226308  2.709074527  2.16543793 -1.65374907 -1.55330643
```

```

## 22  0.774189977 -0.019201196  1.12925215 -0.78455556 -0.43848325
## 23  0.879968694 -0.234866074  0.44008386 -0.91966620 -0.96512862
## 25  0.802274577  0.313638729  0.12649854 -0.35460837 -0.30817925
## 26  1.276715007  0.379080586  1.48431029 -1.36110053 -1.32436953
## 32  0.806601276  0.251042170  1.26551999 -0.54387265 -0.34889925
## 35  0.518249056  0.517949669  0.65252628 -1.03016153 -1.04023440
## 36  0.740232585  0.778626946  1.81101457 -1.31843401 -1.29631797
## 40  0.480968154  0.730458555  1.32942200 -0.71672674 -0.73619172
## 41  0.238290585  0.120163906  1.26594319 -0.80096576 -0.69818638
## 49  0.236952212  0.231441227  1.30847399 -0.79932474 -0.79048505
## 53  0.285785142  0.009120967 -0.18497082 -0.51871035 -0.49368148
## 56  0.935259807  0.190996568  0.69886581 -0.75665823 -0.47739348
## 58  1.029335427  0.942684003  2.46569270 -1.42181826 -1.32798908
## 59  0.997773889  0.672211705  2.76023438 -1.53067258 -1.45105398
## 63  1.095045555 -0.470578858 -0.82272127  0.02337320 -0.13625035
## 65  1.017950991  0.501973483  1.12988694 -0.88848682 -0.69728149
## 78  0.936787017 -0.002483707  0.55751966 -0.38031768 -0.18963879
## 79  0.786346280  0.798129775  2.06852694 -1.25552825 -1.21035352
## 82  0.864269360  0.624517531  2.25325028 -1.26701539 -1.21397308
## 86  0.810425201  0.610116616  2.31863346 -1.39665596 -1.33884775
## 87  0.011418570 -0.367461329 -0.83626342  0.42050000  0.73153815
## 91  0.396131324  0.438488450  0.70309773 -1.10947749 -1.10176685
## 92  0.503276740  0.048649899  0.35205991 -0.33983919 -0.47015437
## 93  0.582363521  0.525449918  1.86052804 -0.95412761 -0.82758550
## 94  -0.332224764  0.228993835 -0.06012916 -0.19324142 -0.12810635
## 101 0.928487685 -0.594049535  0.38528049 -0.37430060 -0.20230724
## 104 0.472196732 -0.143124287 -0.57240316 -0.12760062 -0.18782902
## 106 0.967470469 -0.247838890 -0.11070062 -0.45580459 -0.48372770
## 115 -0.793305063 -1.209031841 -1.32251110  2.26828833  3.22450719
## 118 1.522461157  0.603052427  0.77673315 -1.20192161 -1.21306819
## 123 0.410912444 -0.128042026  0.59052864 -0.69047043 -0.73528683
## 124 1.143703812  0.946695764  2.37132087 -1.28069056 -1.24564419
## 125 0.115025676  0.075173310  0.12417098 -1.10072538 -1.10810107
## 128 0.401904977  0.127124530  1.19907884 -1.37258767 -1.35332597
## 129 0.660489599  0.549089876  1.90686757 -1.15378502 -1.13434285
## 132 0.640843598 -0.395870704  0.04122534 -0.43720636 -0.51177926
## 133 -0.653403723 -0.169380611 -1.12974712  1.83670011  1.87622266
## 138 -0.280075406  0.560378406 -0.06605385 -0.15440395 -0.23307346
## 144 0.222789529  0.023494629  0.89840087 -0.94428149 -0.96150906
## 152 1.051433933  0.253663987  1.49742925 -1.25115220 -1.24564419
## 154 -0.009494991 -0.034365218  0.25641850 -0.07454098 -0.08376679
## 156 0.291674457  0.622604313  2.17474815 -1.29655375 -1.28183975
## 157 0.249325675  0.220338023  1.45849558 -1.00937528 -0.95788951
## 159 0.155006929 -0.400247666  0.11803470 -0.36718952 -0.42400503
##          FAsym    FM3Long   FM3Trans   FAlpha     FDist
## 3      2.04493832 -1.4784975 -2.18297248  1.8891745  0.84261298
## 5     -0.02019987  0.3530767  1.03659284 -0.8810160  2.17636939
## 9      1.93707775  1.4608408  2.06144760 -0.8731243  0.72804326
## 11     -0.49952241  0.9223362  1.12934068 -0.6792662  1.72986595
## 15     0.31469320 -0.8710185  1.37231342 -0.7414185  2.39188678
## 21     -0.88468905  1.2109002  2.86973709 -0.4223057  1.22566283
## 22     -0.91973107  0.9269715 -0.91509215 -0.9822663  1.45500299
## 23     -0.89661857  1.4524154  0.70659067 -0.9809638  1.04667608
## 25     -0.23440936  1.1008044 -0.55664674 -0.8895014  1.98959478

```

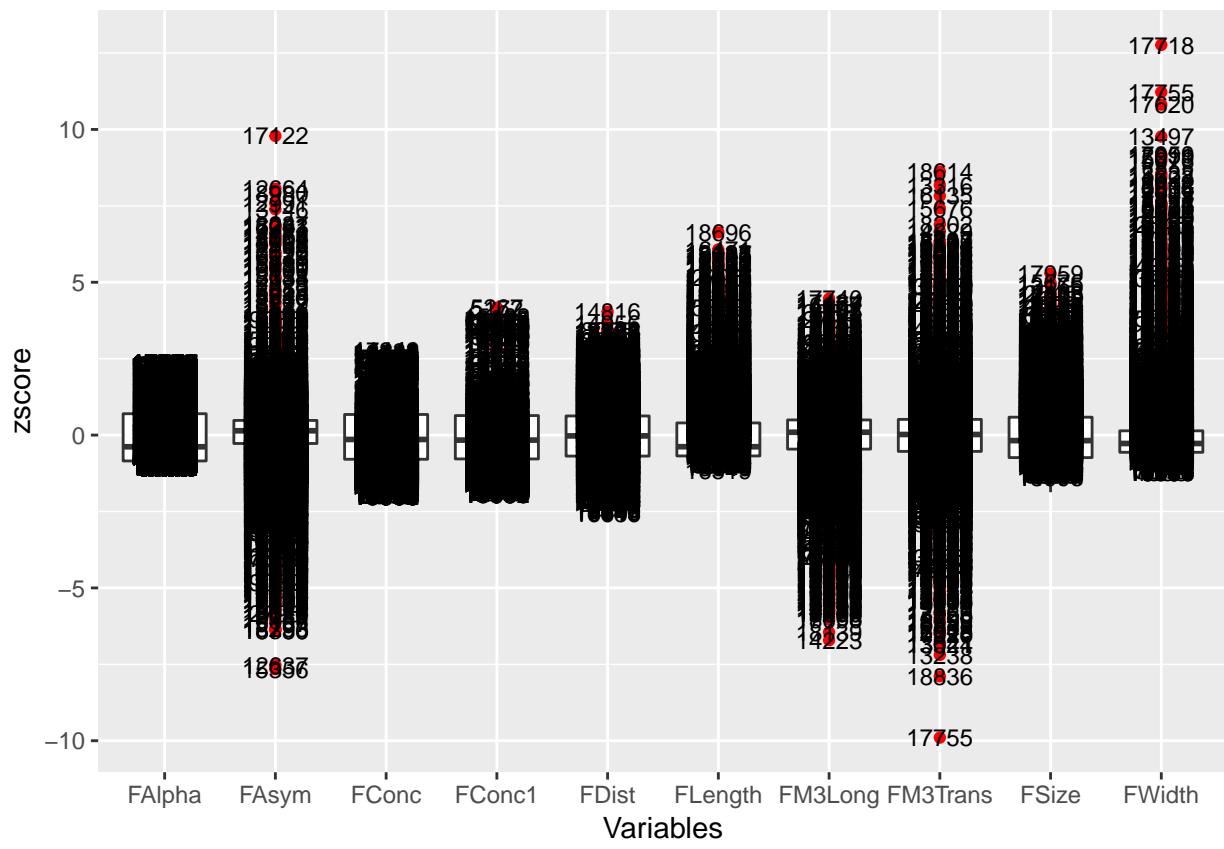
```

## 26 -0.72265835 1.5483897 -0.97456178 -0.9321966 1.33762322
## 32 0.25013731 1.1333122 -1.16886795 -1.0451043 2.38147622
## 35 -0.90856499 0.6379290 0.48360118 -0.9980649 1.19222324
## 36 0.99284166 1.4799428 0.76329471 -0.3857973 0.86681954
## 40 -0.73820067 0.6287506 -1.27350876 -0.8877966 2.38852811
## 41 -0.74834153 -0.5874544 0.80315079 -1.0345579 1.30780994
## 49 -0.38889354 0.8648030 0.93788171 -1.0217129 1.45960612
## 53 -1.43497054 -0.8194088 -0.96067625 -0.8723582 1.05526680
## 56 -0.35759100 0.8088502 -0.92431076 -0.7629442 1.48219356
## 58 -0.92397895 1.0879103 -1.28103729 -0.9130920 1.96941595
## 59 0.43121505 1.5365936 -0.74944529 -1.0558576 0.89735541
## 63 0.89591578 1.6022660 0.35783440 1.7974784 -1.82111163
## 65 -0.51463066 0.9905478 0.97458809 -0.6843383 1.51880449
## 78 0.38818061 0.7446484 0.67374458 -0.9251095 2.21019703
## 79 -0.33315262 0.9888459 1.24437162 -0.9578252 1.50848759
## 82 0.44638749 1.7249500 0.66351288 -0.9177542 -0.47101813
## 86 1.74888925 1.2641511 -0.70268486 -0.9660616 0.82290249
## 87 -0.70058797 -0.9981515 -0.46261213 -0.9868251 -2.15618190
## 91 -1.04839695 0.7434405 -1.21216660 -0.9466007 1.35053607
## 92 -0.54380166 1.0300065 -0.60060798 -0.9047292 2.27039899
## 93 0.03196539 0.9337479 -0.70199346 -1.0497205 1.79540700
## 94 -0.57075836 0.2916926 0.80699668 -0.8101676 1.40977458
## 101 -1.08632888 -1.1659884 0.31360908 -0.9629242 1.32825639
## 104 -0.67212805 0.4715353 0.78942851 -0.1722637 1.18822227
## 106 -1.56638107 0.7844385 -0.73132976 -0.9880509 1.45053367
## 115 0.06357364 -0.1034477 -0.01199024 0.2747241 -0.15792512
## 118 1.74372086 2.3409055 -0.87210559 -0.9601621 -0.06469839
## 123 -1.56485927 0.8227247 -0.61737432 -1.0418366 1.26378583
## 124 -2.59331983 1.1738395 1.05152986 -0.9032466 1.78549154
## 125 -0.75816991 -0.8628930 0.54345011 -0.2854664 -1.33559936
## 128 1.48095553 0.7533719 -0.60656166 -0.9469800 0.53255750
## 129 0.67389122 1.1058613 0.60686646 -0.8999022 1.35901974
## 132 0.68253898 1.2844138 -0.52088622 -0.8650029 1.05442378
## 133 0.35534444 0.2872239 -0.72622592 1.7875794 1.64122897
## 138 -0.27305911 0.3479826 1.34194003 -0.6902378 1.53288150
## 144 -1.63127308 -0.7058581 0.78317714 -0.9395136 0.48105331
## 152 0.57487264 1.6629149 0.56493139 -0.9359969 0.83788942
## 154 -0.82656494 0.3412670 -1.00914116 -0.8258895 1.54959460
## 156 -1.34564523 0.6462015 -1.16183396 -1.0300873 0.70281972
## 157 0.82556150 0.7060112 0.86957278 -0.9855609 1.21546637
## 159 -1.79235455 0.8517030 0.57239270 -0.8585287 1.28082008

cat("media de la desviación por columna:\n",
    apply(mydata.numeric.scaled[indices.MCD.outliers,],2,mean),
    "\n")

## media de la desviación por columna:
## 1.033319 0.7869901 0.9005836 -0.774348 -0.7177835 -0.3811679 0.08248971 0.01057504 -0.1080192 0.678
MiBoxPlot_juntos(mydata.numeric.scaled,is.MCD.outlier)

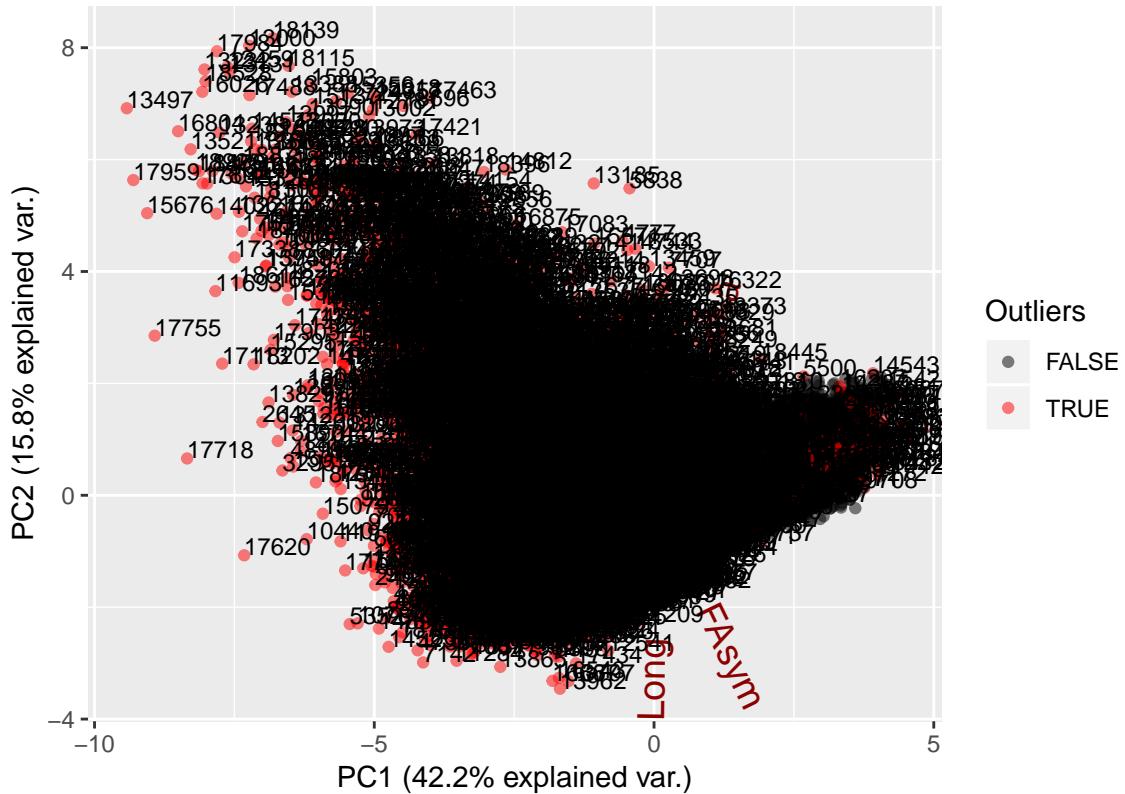
```



Como se puede ver, la gráfica con los boxplots no aporta mucha información ya que hay un gran número de outliers multivariados (6574); vamos a utilizar un biplot para intentar obtener más información de los outliers.

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

Outliers MAGIC



Por desgracia, el biplot tampoco nos da demasiada información, ya que hay demasiados puntos y tapan las variables; además, no explica bien la varianza de los datos, ya que $PC1 + PC2 = 42.2 + 15.8 = 58$. Ahora vamos a ver cual de los datos son realmente multivariantes puros, es decir, no hay ningún valor en alguna de sus columnas que sea un outlier univariado, para ello haremos los siguientes.

```
indices.de.outliers.en.alguna.columna = vector_claves_outliers_IQR_en_alguna_columna(mydata.numeric)
indices.de.outliers.MCD.pero.no.1variantes = setdiff(indices.MCD.outliers, indices.de.outliers.en.alguna
nombres.de.outliers.multivariantes.MCD.pero.no.1variantes = rownames(mydata.numeric)[indices.de.outlier

#Mostramos una porción de los datos que nos salen.
head(indices.MCD.outliers,10)

##  3   5   9  11  15  21  22  23  25  26
##  3   5   9  11  15  21  22  23  25  26
head(indices.de.outliers.MCD.pero.no.1variantes,10)

## [1]  5 11 15 22 23 25 26 32 35 36
head(nombres.de.outliers.multivariantes.MCD.pero.no.1variantes,10)

## [1] "5"  "11" "15" "22" "23" "25" "26" "32" "35" "36"
cat("número de datos multivariantes puros:",
  length(indices.de.outliers.MCD.pero.no.1variantes),
  "\n",
  "número de datos multivariantes:",
  length(indices.MCD.outliers),"\n")

## número de datos multivariantes puros: 3647
```

```
## número de datos multivariantes: 6574
```

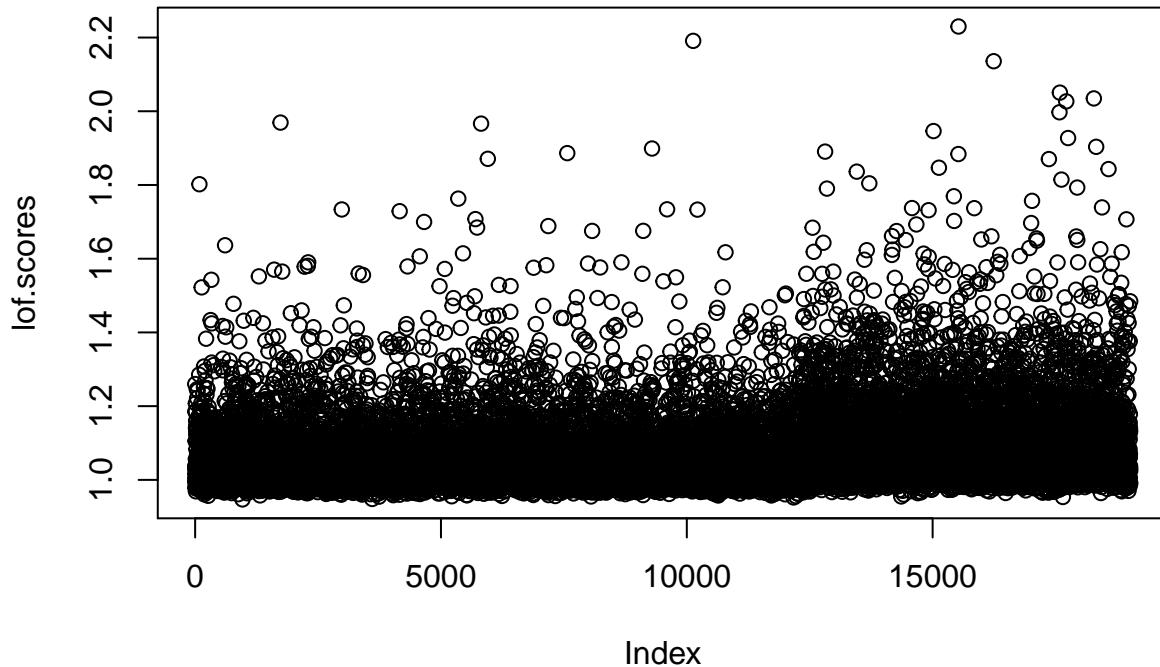
Como podemos ver solamente el 50% más o menos son realmente outliers multivariantes puros.

Cálculo de LOF outliers

En este apartado, identificadores outliers mediante métodos centrados en ratio de densidad; para ello utilizaremos los datos normalizados.

```
numero.de.vecinos.lof = 15  
lof.scores = lofactor(mydata.numeric.scaled,numero.de.vecinos.lof)  
head(lof.scores,10)
```

```
## [1] 0.9963571 1.0366289 1.2599699 1.0288543 1.0163234 0.9784905 0.9965702  
## [8] 1.0060435 1.1842516 1.1866542  
plot(lof.scores)
```



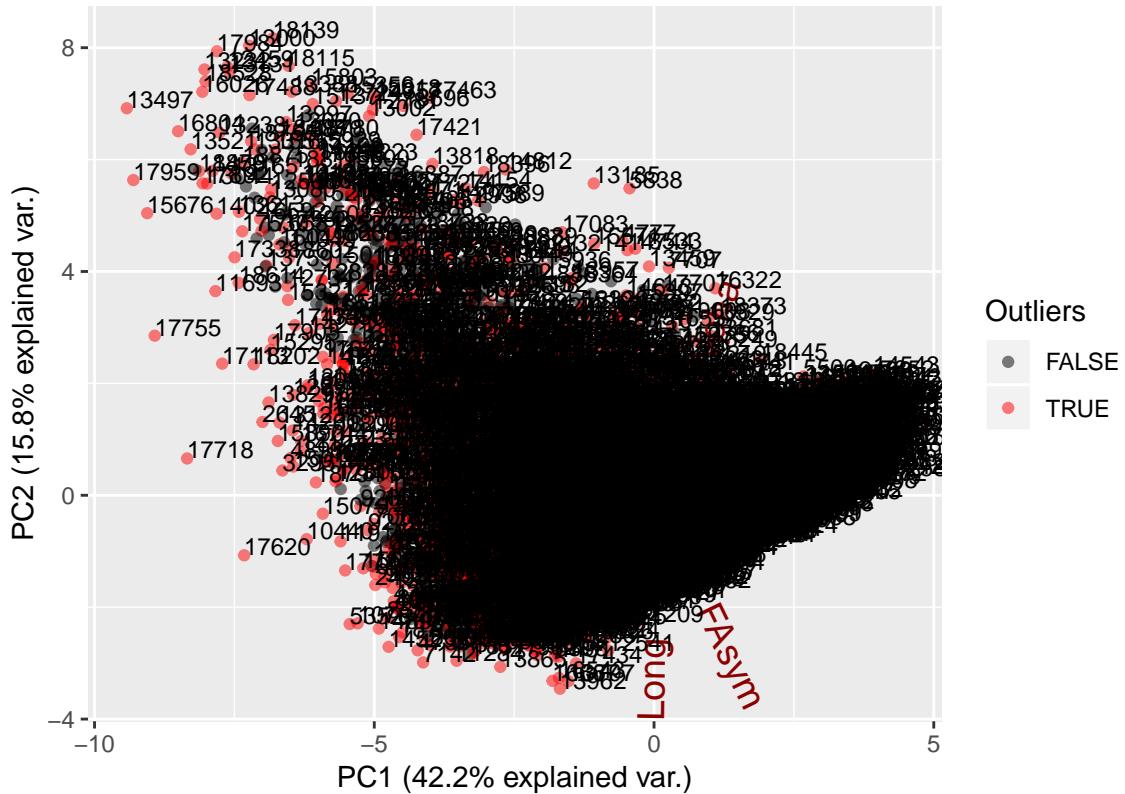
```
numero.de.outliers = 6000
```

```
indices.de.lof.outliers.ordenados = order(lof.scores, decreasing = TRUE)[1:numero.de.outliers] # decrea  
head(indices.de.lof.outliers.ordenados,10)
```

```
## [1] 15526 10130 16242 17588 18279 17718 17580 1737 5816 15018  
is.lof.outlier = rownames(mydata.numeric.scaled) %in% indices.de.lof.outliers.ordenados
```

```
MiBiPlot_Multivariate_Outliers(mydata.numeric.scaled,is.lof.outlier, "LOF outliers")
```

LOF outliers



En la gráfica se puede ver que la mayoría de los puntos de alrededor son detectados como outliers. Lo siguiente que vamos a hacer es descubrir cuales de los puntos son univariados y cuales son multivariados.

```
vector.claves.outliers.IQR.en.alguna.columna = vector_claves_outliers_IQR_en_alguna_columna(mydata.numeric)
vector.es.outlier.IQR.en.alguna.columna = vector_es_outlier_IQR_en_alguna_columna(mydata.numeric.scaled)

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

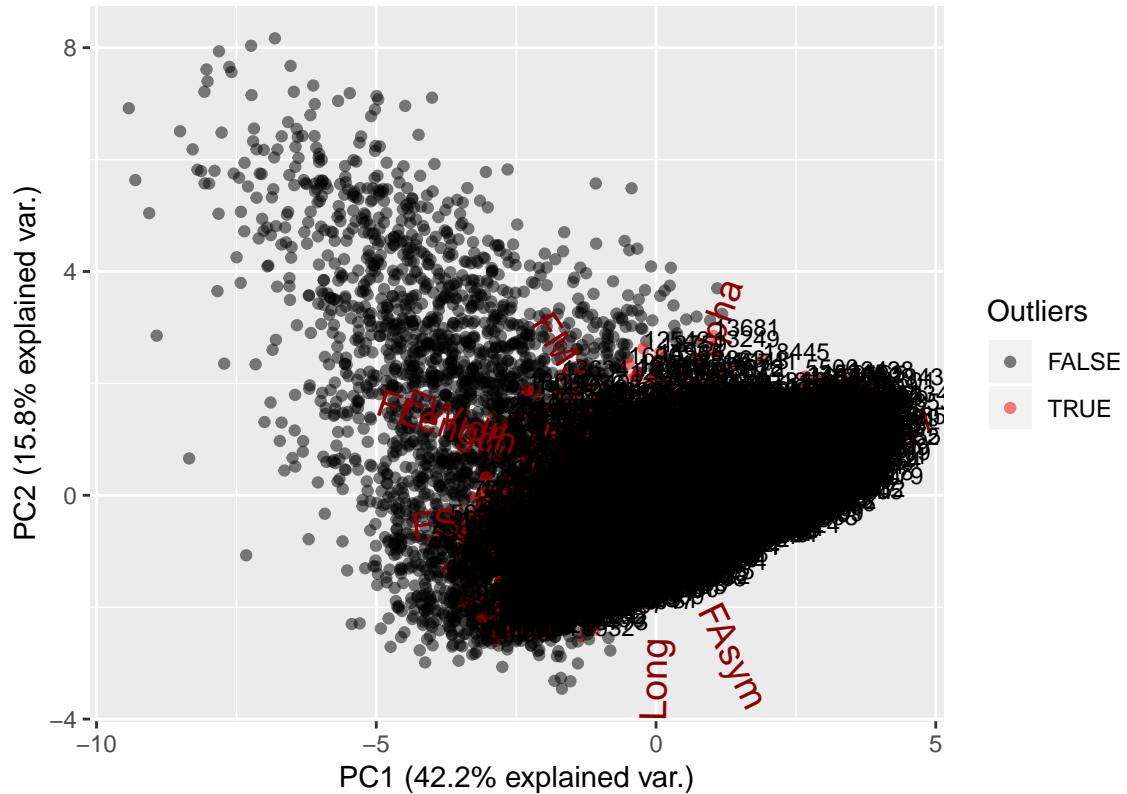
head(indices.de.outliers.multivariantes.LOF.pero.no.1variantes,10)
```

```
## [1] 10130 16242 17588 17580 1737 5816 15018 18326 12813 7569
```

Vamos a representar los datos en un biplot.

```
is.lof.outlier = rownames(mydata.numeric.scaled) %in% indices.de.outliers.multivariantes.LOF.pero.no.1va
MiBiPlot_Multivariate_Outliers(mydata.numeric.scaled,
                                is.lof.outlier, "LOF outliers")
```

LOF outliers



Como se puede ver , los puntos que se representan están en la parte central, en vez de estar en los extremos, de ahí podemos saber que el resto de los datos que antes estaban representados como LOF outlier eran outliers univariantes.

Cálculo de outliers basados en clustering

En este apartado se utilizarán diferentes métodos de clústering para detección de outliers, para utilizar estos datos utilizaremos los datos escalados (esto normalmente es requerido por los algoritmos de clústering). Lo primero que vamos a hacer es calcular outliers dependiendo de la distancia euclídea que tenga hacia su clúster más cercano; para ello hay que construir un modelo k-means. Después calculamos la distancia de cada punto a su centroide y nos quedamos con los que tengan la distancia más alta.

```

numero.de.outliers    = 15
numero.de.clusters   = 3

set.seed(2)

# Cálculamos el modelo k-means.
modelo.kmeans = kmeans(mydata.numeric.scaled, centers=numero.de.clusters)

indices.clusterings.magic = modelo.kmeans$cluster
head(indices.clusterings.magic)

## 1 2 3 4 5 6
## 1 1 3 1 2 2

```

```

centroides.normalizados.magic = modelo.kmeans$centers
head(centroides.normalizados.magic)

##      FLength      FWidth      FSize      FConc      FConc1      FAsym
## 1 -0.6118259 -0.5133722 -0.7378330  0.7769070  0.7500581  0.1285573
## 2  0.3099416  0.1779358  0.6209681 -0.7378236 -0.7174656  0.1312237
## 3  2.4273983  2.5146273  1.4891445 -1.0794632 -1.0122514 -1.6524143
##      FM3Long      FM3Trans      FAlpha      FDist
## 1 -0.1449232 -0.01157009  0.3211732 -0.4580005
## 2  0.4630128 -0.01190182 -0.4893117  0.3857352
## 3 -1.6917170  0.14925028  0.6259908  0.9227557

# Función para calcular la distancia euclídea de cada punto a los centros.
distancias_a_centroides = function (datos.normalizados,
                                      indices.asignacion.clustering,
                                      datos.centroides.normalizados){

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

# Obtenemos los elementos con mayor distancia.
top.outliers.magic = distancias_a_centroides(mydata.numeric.scaled,
                                              indices.clusterings.magic,
                                              centroides.normalizados.magic)
top.outliers.magic = order(top.outliers.magic,decreasing = TRUE)[1:numero.de.outliers]
top.outliers.magic

## [1] 17755 17718 17122 17620 12991 18237 15019 12566 18836 18008 18614
## [12] 12664 18202 18980 15676

```

Este proceso se puede automatizar con una función, veamos como se haría.

```

# Función para calcular outliers con k-means.
top_clusterings_outliers = function(datos.normalizados=mydata.numeric.scaled,
                                      indices.asignacion.clustering=indices.asignacion.clustering,
                                      datos.centroides.normalizados=centroides.normalizados.magic,
                                      numero.de.outliers=numero.de.outliers){

  top.outliers.magic = distancias_a_centroides(datos.normalizados,
                                                indices.asignacion.clustering,
                                                centroides.normalizados.magic)

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

  mi.lista = list(indices=indices.top.outliers.magic,
                  distancias=top.outliers.magic[indices.top.outliers.magic]
                 )
  mi.lista
}

top_clustering = top_clusterings_outliers(mydata.numeric.scaled,
                                           indices.clusterings.magic,
                                           centroides.normalizados.magic,
                                           numero.de.outliers)

top_clustering$indices

```

```

## [1] 17755 17718 17122 17620 12991 18237 15019 12566 18836 18008 18614
## [12] 12664 18202 18980 15676
top_clustering$distancias

##    17755    17718    17122    17620    12991    18237    15019    12566
## 14.83511 13.49707 13.36778 13.24199 12.66690 12.58267 12.03547 11.67616
##    18836    18008    18614    12664    18202    18980    15676
## 11.62340 11.34547 11.31880 11.20261 11.13639 11.11494 10.87197

```

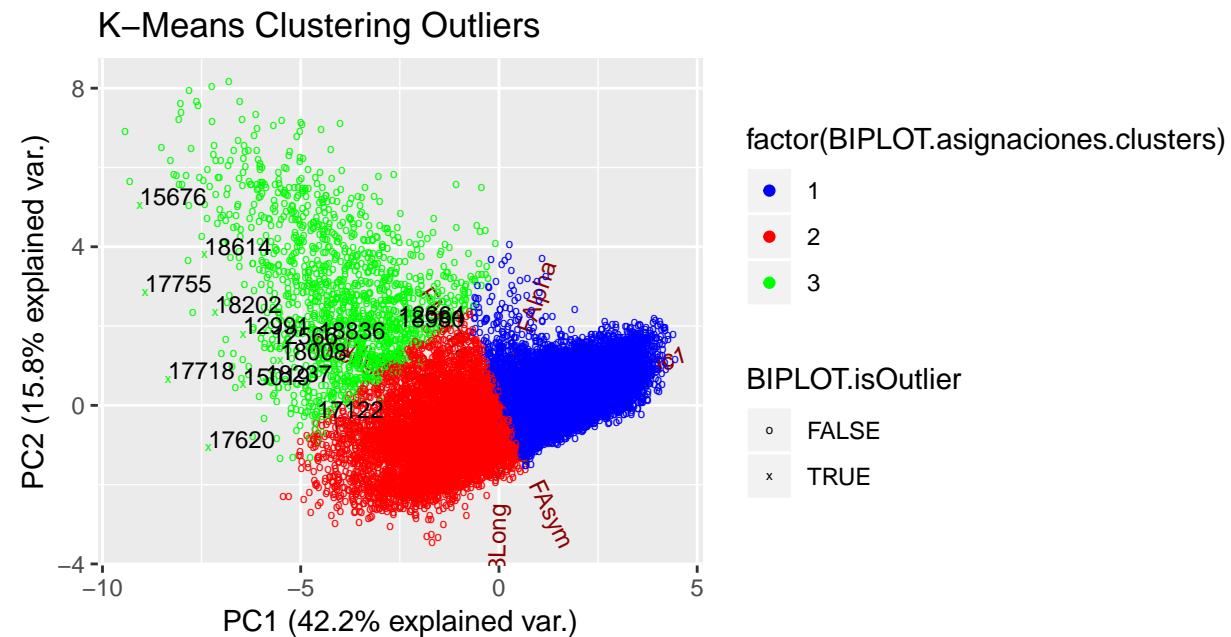
A continuación mostaremos los datos en un biplot para intentar identificarlos, y así también descubrir si los datos que estamos describiendo como outliers lo son realmente o no.

```

numero.de.datos      = nrow(mydata.numeric.scaled)
is.kmeans.outlier   = rep(FALSE, numero.de.datos)
is.kmeans.outlier[top_clustering$indices] = TRUE

BIPLOT.isOutlier      = is.kmeans.outlier
BIPLOT.cluster.colors = c("blue","red","green")      # Tantos colores como diga numero.de.clusters
BIPLOT.asignaciones.clusters = indices.clusterings.magic
MiBiPlot_Clustering_Outliers(mydata.numeric.scaled, "K-Means Clustering Outliers")

```



Como se puede ver en la gráfica, la mayoría de los outliers son aquellos que están en la periferia del tercer clúster; otros sin embargo parecen que son considerados outliers del segundo clúster y están mezclados con los datos del tercer clúster, por lo que no se pueden diferenciar a simple vista. Lo siguiente que vamos a hacer es obtener la posición real de los centroides obtenidos por k-means, para ellos debemos deshacer la normalización z-score; para obtener dichos datos haremos lo siguiente.

```

# Obtenemos medias y desviaciones de los datos.
mis.datos.medias = colMeans(mydata.numeric)
mis.datos.desviaciones = apply(mydata.numeric, 2, sd)

# Deshacemos el z-score de cada columna.
valores.centroides = sweep(centroides.normalizados.magic, 2, mis.datos.desviaciones, '*')
valores.centroides = sweep(valores.centroides, 2, mis.datos.medias, '+')

```

Lo siguiente que vamos a hacer es utilizar la distancia de mahalanobis en vez de la distancia euclídea para hacer el cálculo de los outliers, después analizaremos los resultados obtenidos. Para ello utilizaremos la siguiente función.

```

library(MASS)

## 
## Attaching package: 'MASS'

## The following object is masked from 'package:EnvStats':
## 
##     boxcox

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(mydata.numeric[seleccion[,x],])$cov)
  lista.vector.de.medias          = lapply(1:k, function(x) cov.rob(mydata.numeric[seleccion[,x],])$center)

  mah.distances      = lapply(1:k,
                               function(x) mahalanobis(mydata.numeric[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)
}

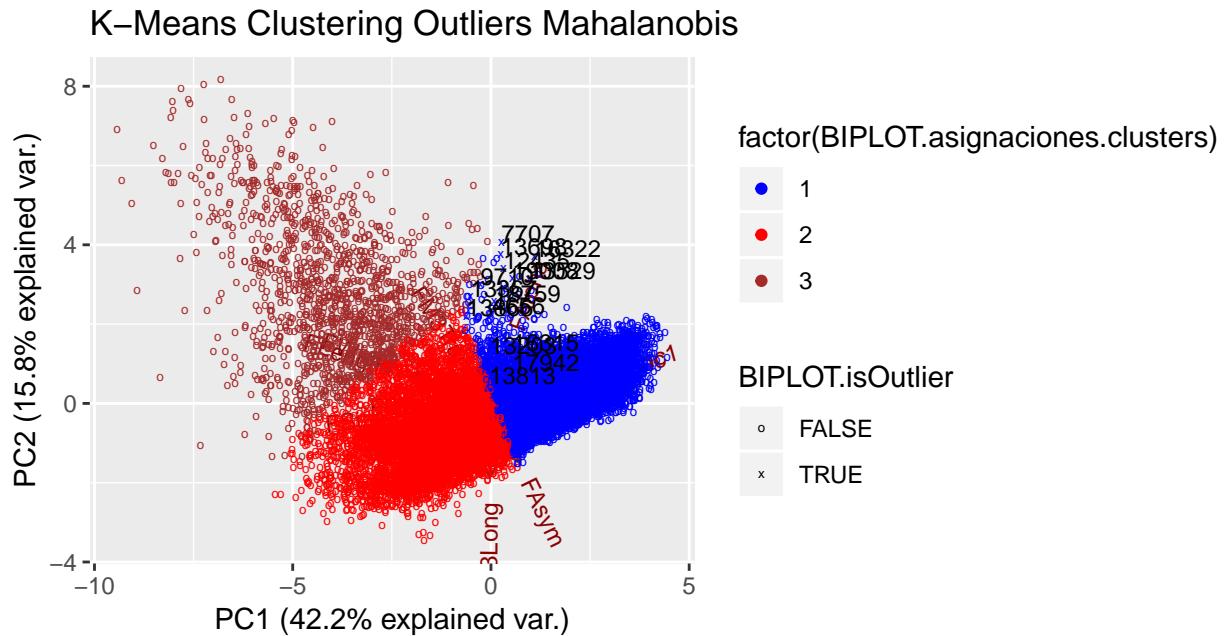
top.clustering.outliers.mah = top_clustering_outliers_distancia_mahalanobis(mydata.numeric,
                           indices.clusterings.magic,
                           numero.de.outliers)

numero.de.datos = nrow(mydata.numeric)
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.clusters
BIPLOT.asignaciones.clusters = indices.clusterings.magic

```

```
MiBiPlot_Clustering_Outliers(mydata.numeric, "K-Means Clustering Outliers Mahalanobis")
```



Como se puede ver, hay una clara diferencia en los outliers obtenidos por este método. En este caso la mayoría de los outliers pertenecen al primer clúster, y el resto (que están mezclados con los datos del primer clúster) pertenecen al segundo clúster; además no se repite ninguno de los que teníamos en los outliers calculados con k-means y distancia de Mahalanobis. Esto se debe a que Mahalanobis tiene una forma elíptica para el cálculo de distancias, por ello ninguno de los datos del tercer clúster son outliers aunque estén más dispersos que el resto de clúster; por esa misma razón sí que aparecen outliers en el primer y segundo clúster, ya que no se encuentran dentro del “radio” de la elipse producida en su correspondiente clúster.

Por último, crearemos una función que calcule los outliers utilizando la distancia relativa en vez de la distancia, al igual que los métodos basados en LOF outliers. De esta manera podemos detectar outliers que se encuentren cerca de datos que sí pertenecen a un clúster y que con solamente la distancia no seríamos capaces de detectarlos. Para ello utilizamos la siguiente función.

```
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.clust
  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.asignac
}
```

```

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(mydata.numeric.scal
                                         indices.clusterings
                                         centroides.normalizadas
                                         numero.de.outliers)

cat("Indices de los top k clustering outliers (k-means, usando distancia relativa)\n")

## Indices de los top k clustering outliers (k-means, usando distancia relativa)
top.outliers.kmeans.distancia.relativa$indices

## [1] 12518 14525 15999 12548 12541 18903 15524 17980 14849 12746 13865
## [12] 14896 19014 17249 17532
cat("Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa)\n")

## Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa)
top.outliers.kmeans.distancia.relativa$distancias

##    12518    14525    15999    12548    12541    18903    15524    17980
## 5.014512 5.001679 4.925815 4.651340 4.511819 4.452993 4.449074 4.283036
##    14849    12746    13865    14896    19014    17249    17532
## 4.260999 4.256849 4.247740 4.171221 4.064839 4.059198 4.059198

```

Como se puede ver los outliers seleccionados en este caso son diferentes a los que selecciona k-means utilizando la distancia en vez de la distancia relativa. Vamos a pintarlos con un biplot para ver donde aparecen estos outliers calculados.

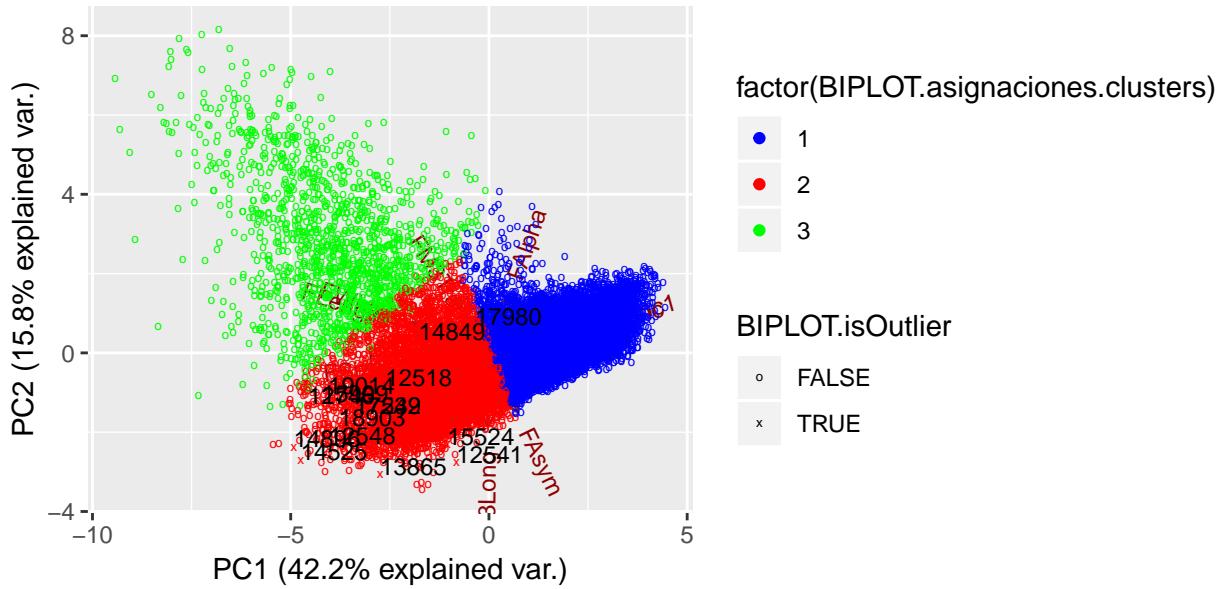
```

numero.de.datos = nrow(mydata.numeric)
is.kmeans.outlier.rel = rep(FALSE, numero.de.datos)
is.kmeans.outlier.rel[top.outliers.kmeans.distancia.relativa$indices] = TRUE

BIPLOT.isOutlier = is.kmeans.outlier.rel
BIPLOT.cluster.colors = c("blue","red","green") # Tantos colores como diga numero.de.clusters
BIPLOT.asignaciones.clusters = indices.clusterings.magic
MiBiPlot_Clustering_Outliers(mydata.numeric, "K-Means Clustering Outliers Mahalanobis")

```

K-Means Clustering Outliers Mahalanobis



Como se puede ver, en este caso los outliers seleccionados son datos que se encuentran relativamente cerca de los clústeres pero que su distancia relativa es más alta que para el resto. Dentro de la gráfica podemos distinguir fácilmente los seleccionados del segundo clúster; los demás outliers seleccionados que se encuentran mezclados con los datos del segundo clúster posiblemente serán del primer o tercero clúster (a simple vista no se pueden distinguir por la gran cantidad de datos), exceptuando el 17980 que seguramente pertenezca también al segundo clúster pero está mezclado con los datos del primer clúster.

Resumen final

Por todas las pruebas que se han podido realizar podemos decir lo siguiente:

1. El cálculo de outliers con IQR es poco útil en este caso ya que obtenemos un número muy alto de outliers, unos 3000 outliers en alguna columna, por lo que es difícil saber cual de estos outliers realmente nos puede dar alguna información relevante del dataset.

2. El cálculo de outliers con test estadísticos puede ser más útil, aunque el número de outliers que se han seleccionado sea muy pequeño para el tamaño del dataset, los outliers que podemos seleccionar con el test de Rosner al representarlos de verdad parecen outliers. El test de Grubbs en este dataset es poco útil, porque es bastante fácil que haya más de un outlier por columna con tantos datos y tengamos problemas de masking al realizar este test. Esto se ve reflejado en la diferencia de resultados obtenidos entre el test de Rosner y el test de Grubbs.
3. Con la detección de outliers multivariados tenemos el mismo problema que con los outliers obtenidos con IQR, obtenemos un número muy alto de ellos, unos 3600, por lo que también es complicado obtener alguna información relevante de ellos.
4. El cálculo de LOF outliers nos muestra que muchos de los outliers de nuestro dataset son realmente univariados y se encuentran en los extremos del biplot que los representa; aún así también se detectan algunos multivariados que se encuentran en la parte central de los datos más o menos. Aún así, la calidad del biplot que podemos obtener es de poca calidad y no es muy representativo.
5. El cálculo de outlier mediante técnicas de clústering obtiene unos resultados interesantes dependiendo de la técnica que utilizemos. Para este estudio hemos escogido un número relativamente pequeño de outliers, 15 solamente. Si utilizamos k-means con distancia euclídea, la mayoría de los outliers seleccionados son muy parecidos a los obtenidos con LOF outliers, los cuales también son outliers

1-variados; si utilizamos la distancia de Mahalanobis o la distancia relativa, obtenemos unos resultados más parecidos a los obtenidos por LOF outliers con outliers únicamente multivariados.