

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.

```
# 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    FWidth   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.primero - (1.5*iqr)
extremo.inferior.outlier.extremo = cuartil.primero - (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.inferior.outlier.extremo
vector.es.outlier.extremo = columna.scaled < extremo.inferior.outlier.extremo | columna.scaled > extremo.superior.outlier.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("índices 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")

## índices 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("índices 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")

## índices 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
```

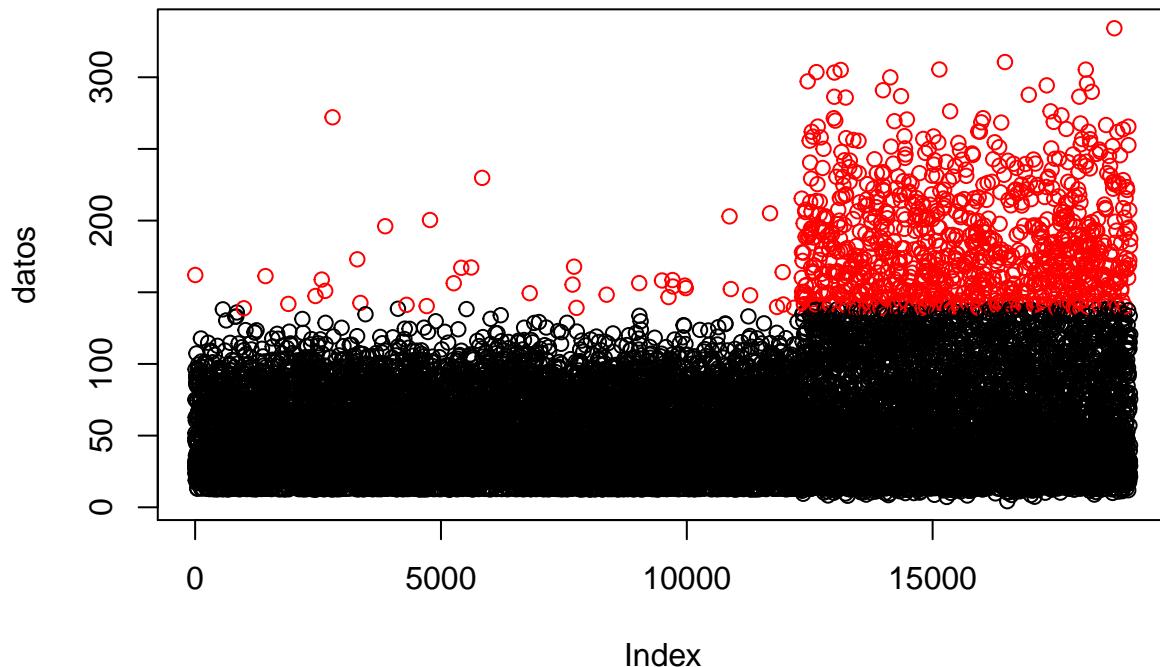
```
valores.normalizados.outliers.extremos = columna.scaled[claves.outliers.extremos]
sd(valores.normalizados.outliers.extremos)
```

```
## [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()*.

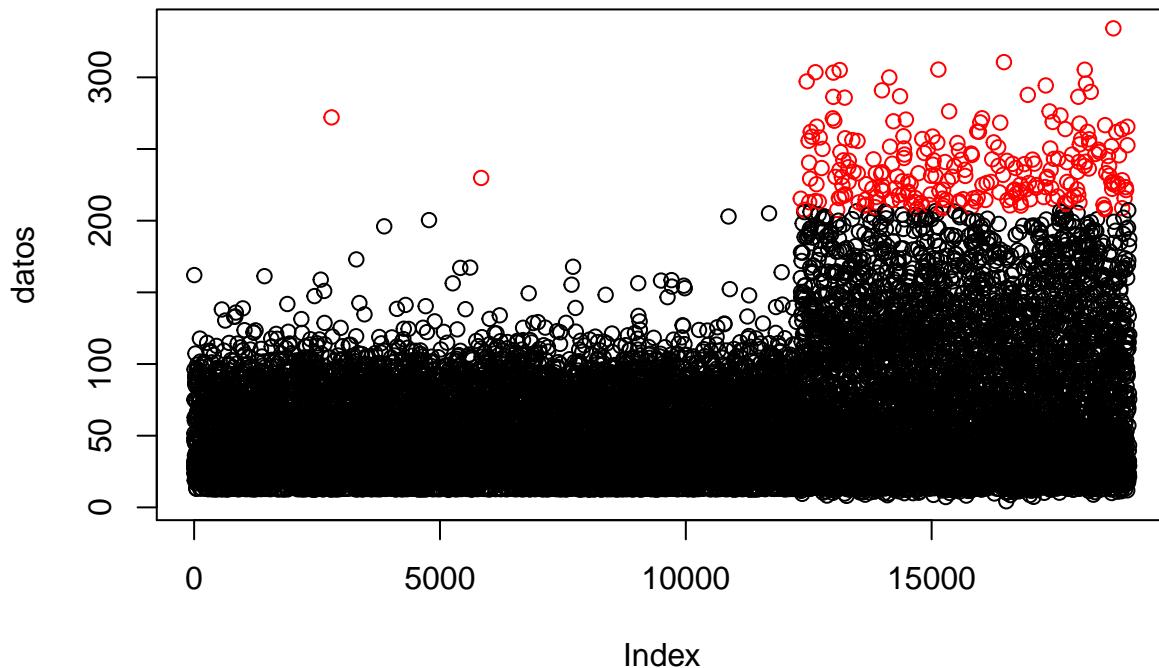
```
MiPlot_Univariate_Outliers(columna,claves.outliers.normales, "Outliers normales")
```

Outliers normales



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

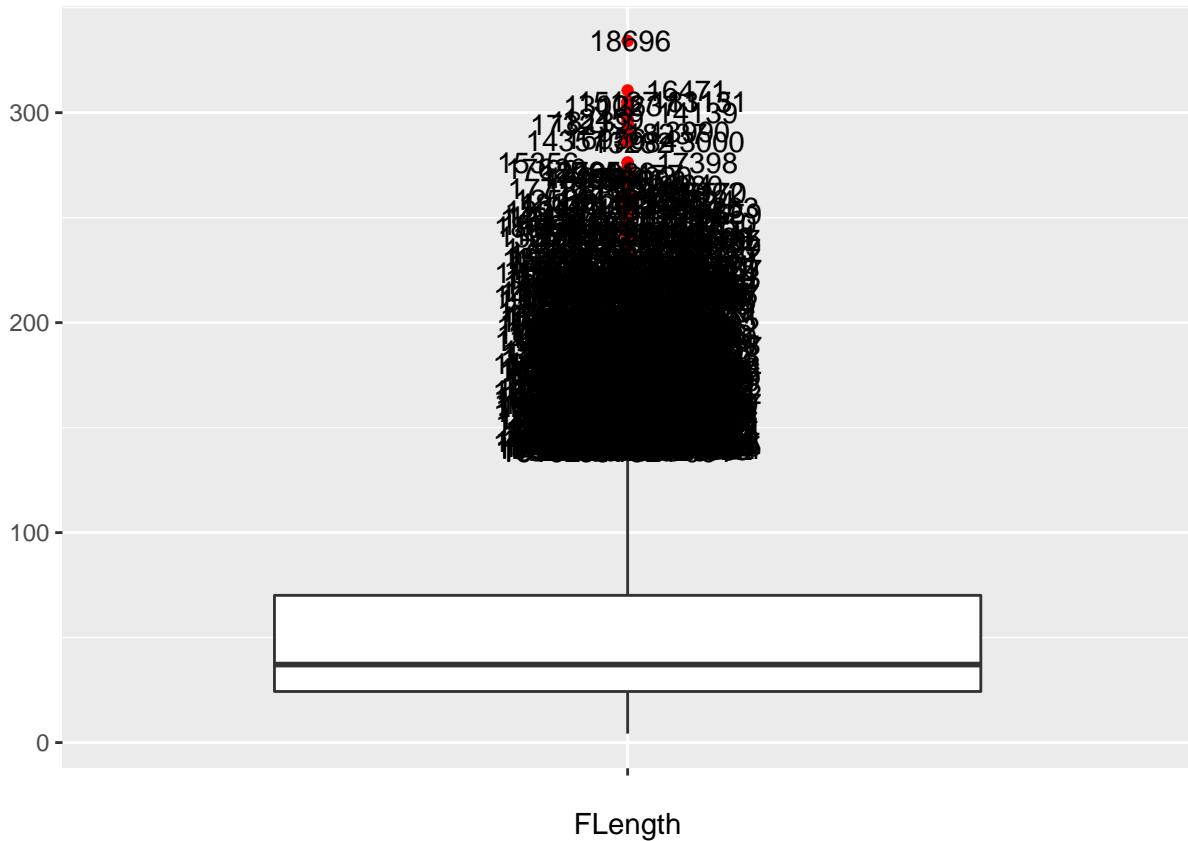
Outliers extremos



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()*.

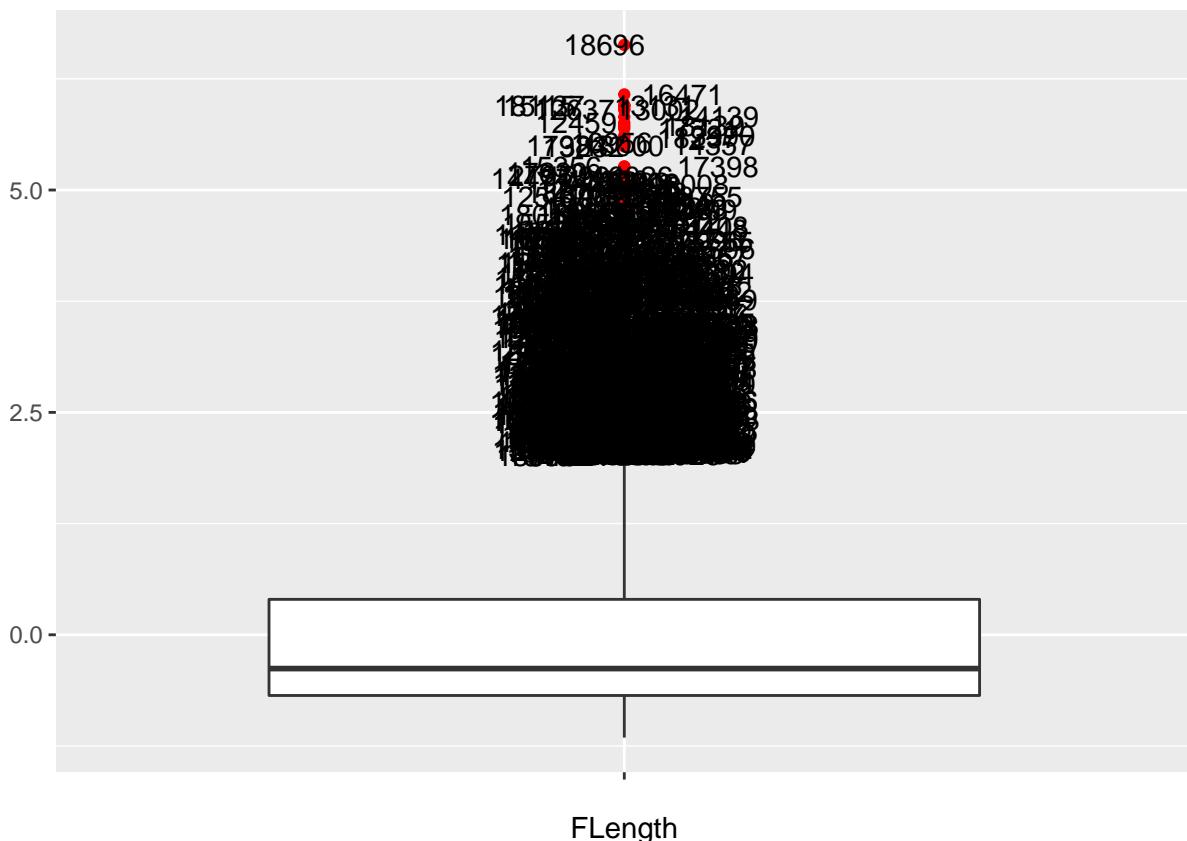
```
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric,indice.columna,coef=1.5)
```

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



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

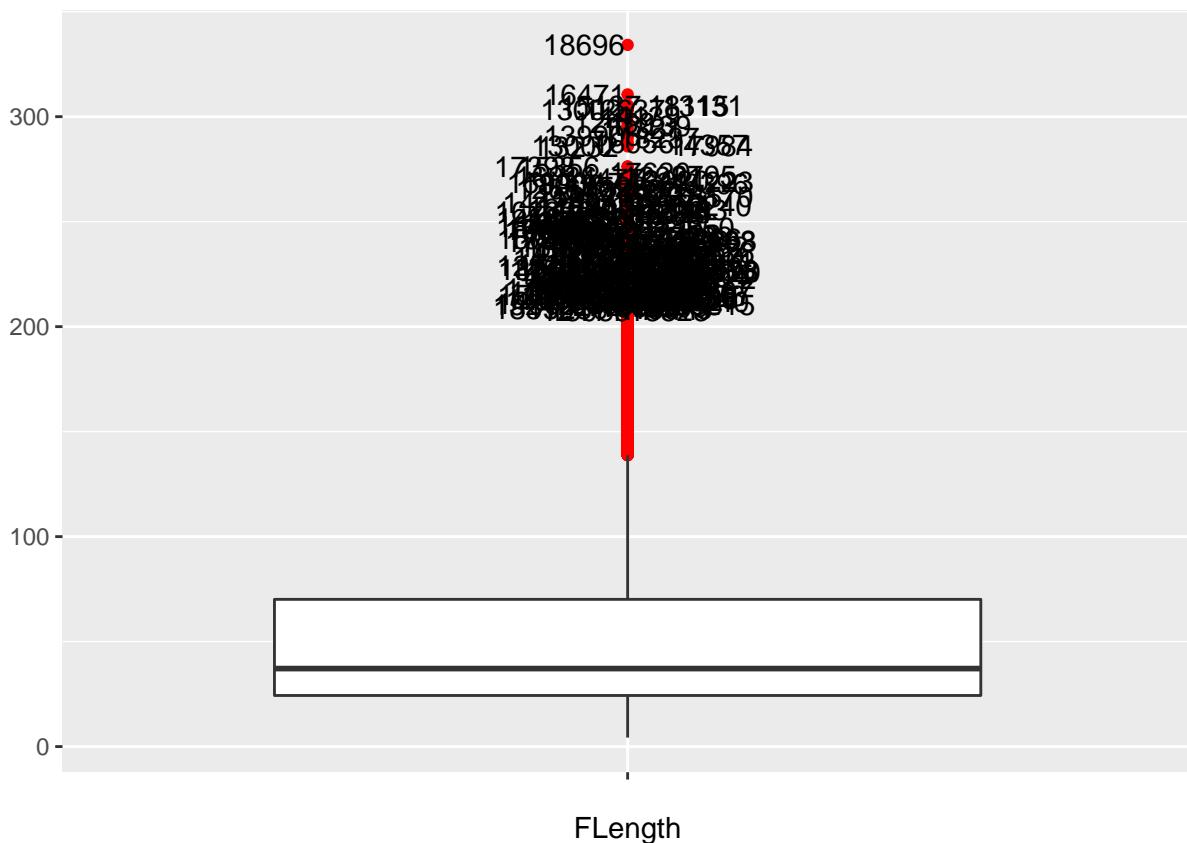
```
## [1] 971
```



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.

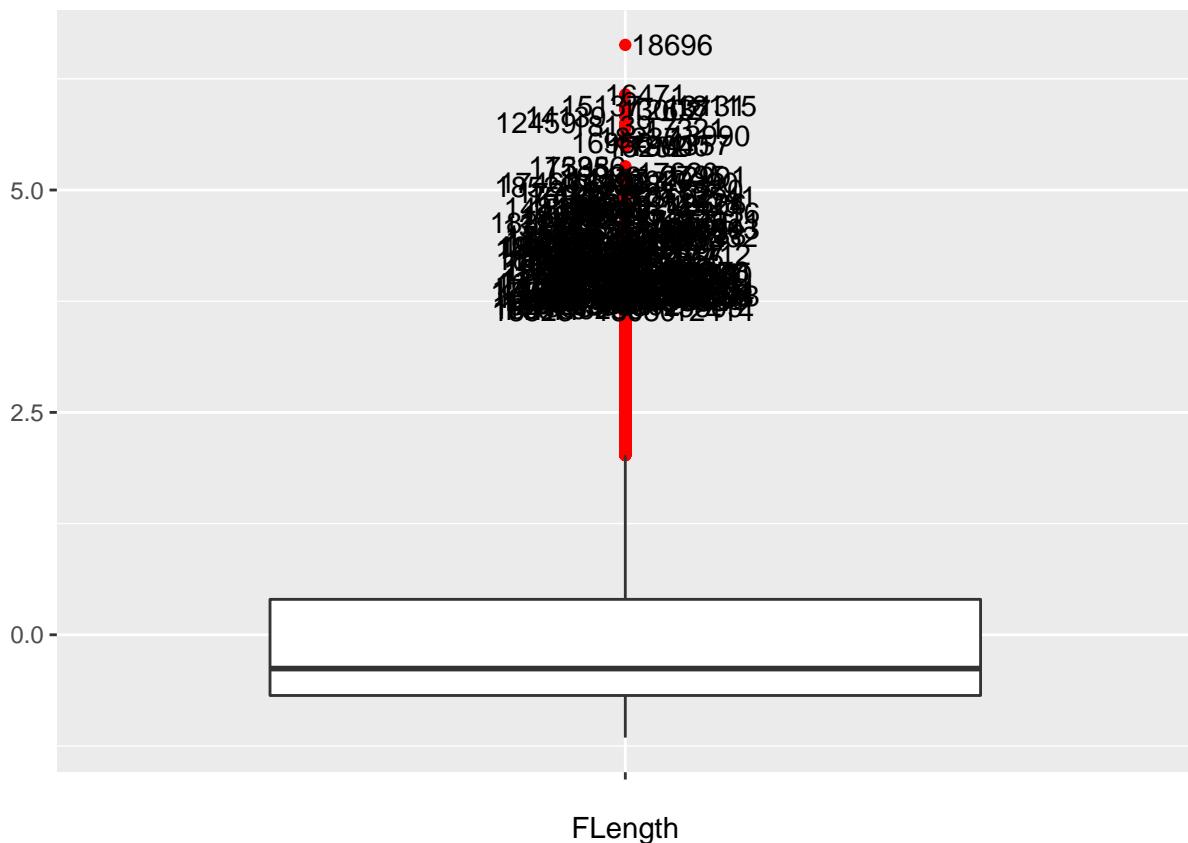
```
MiBoxPlot IQR Univariate Outliers(mydata.numeric,indice.columna,coef=3)
```

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



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

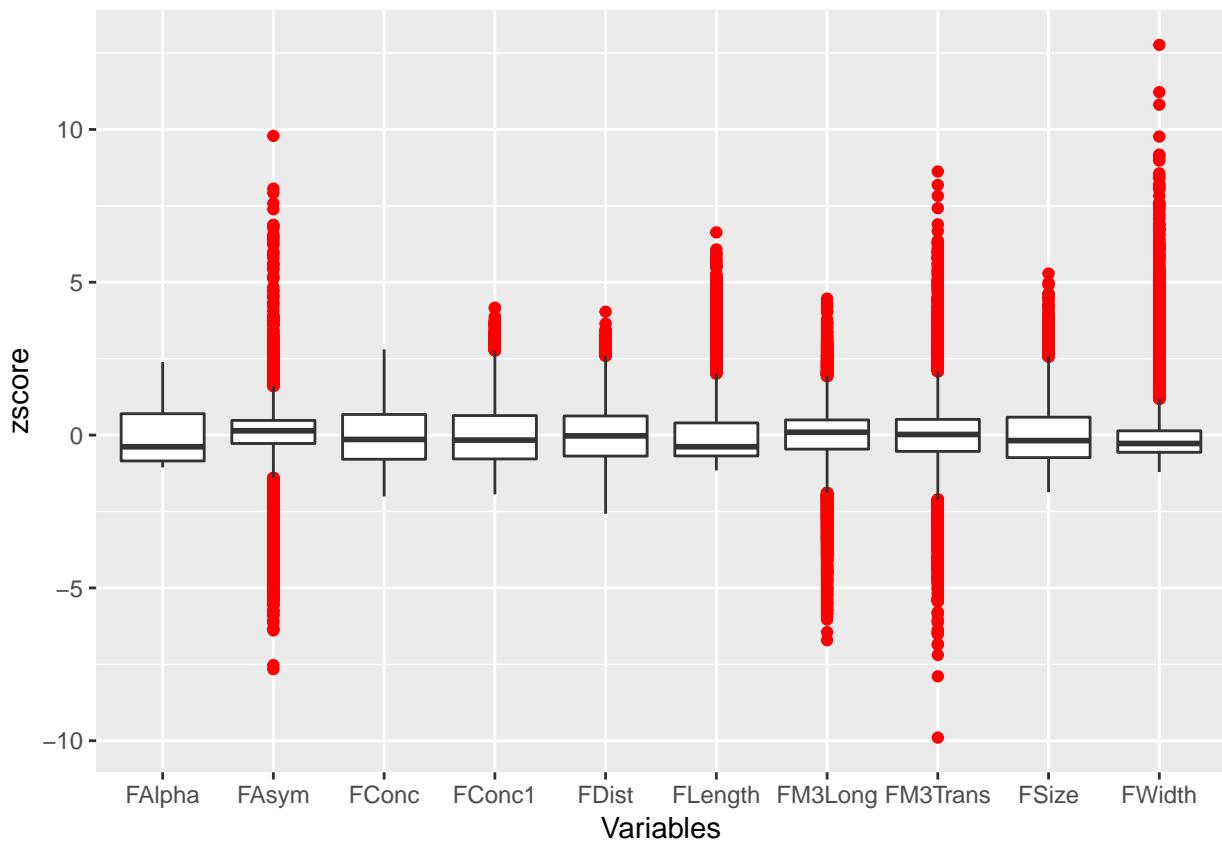
```
## [1] 236
```



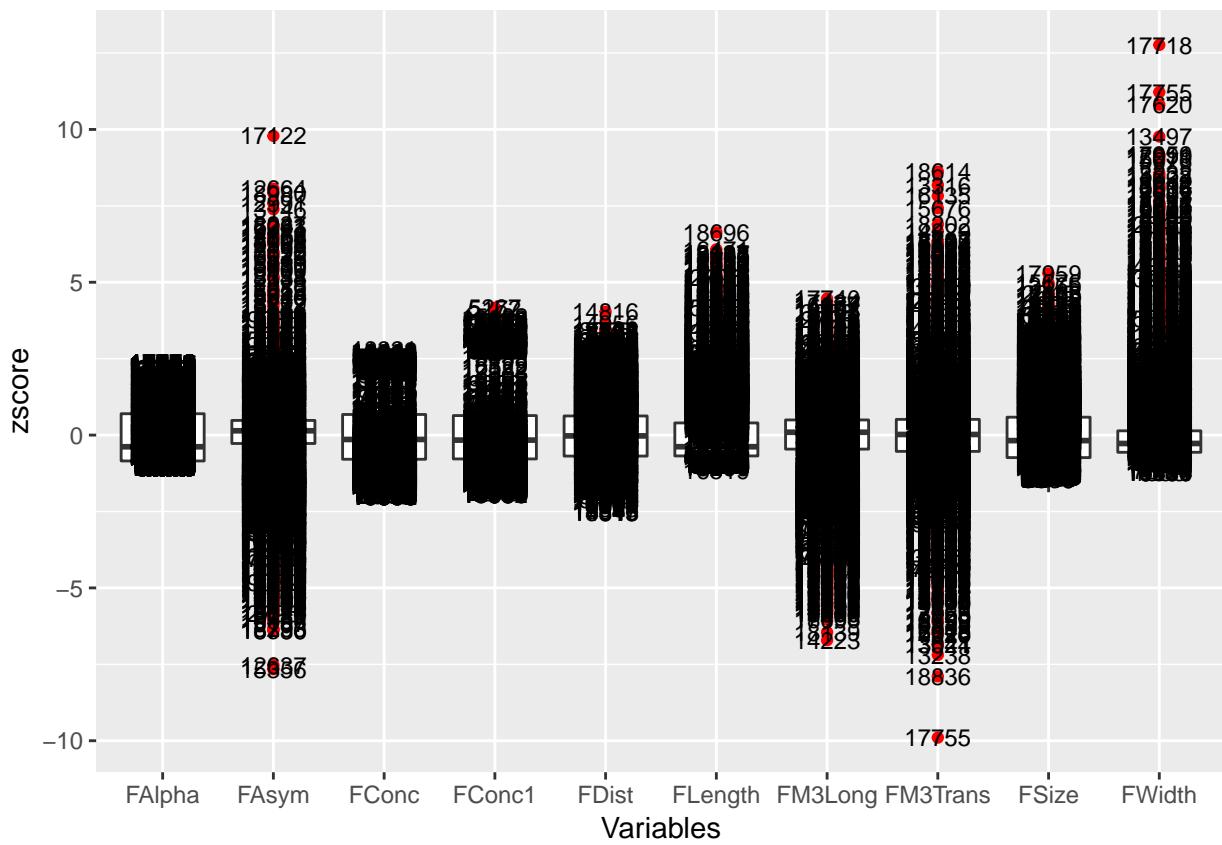
Como se puede ver, los outliers que se consideran extremos son aquellos que tiene valores superiores a 200 para la columna que estamos analizando.

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()*.

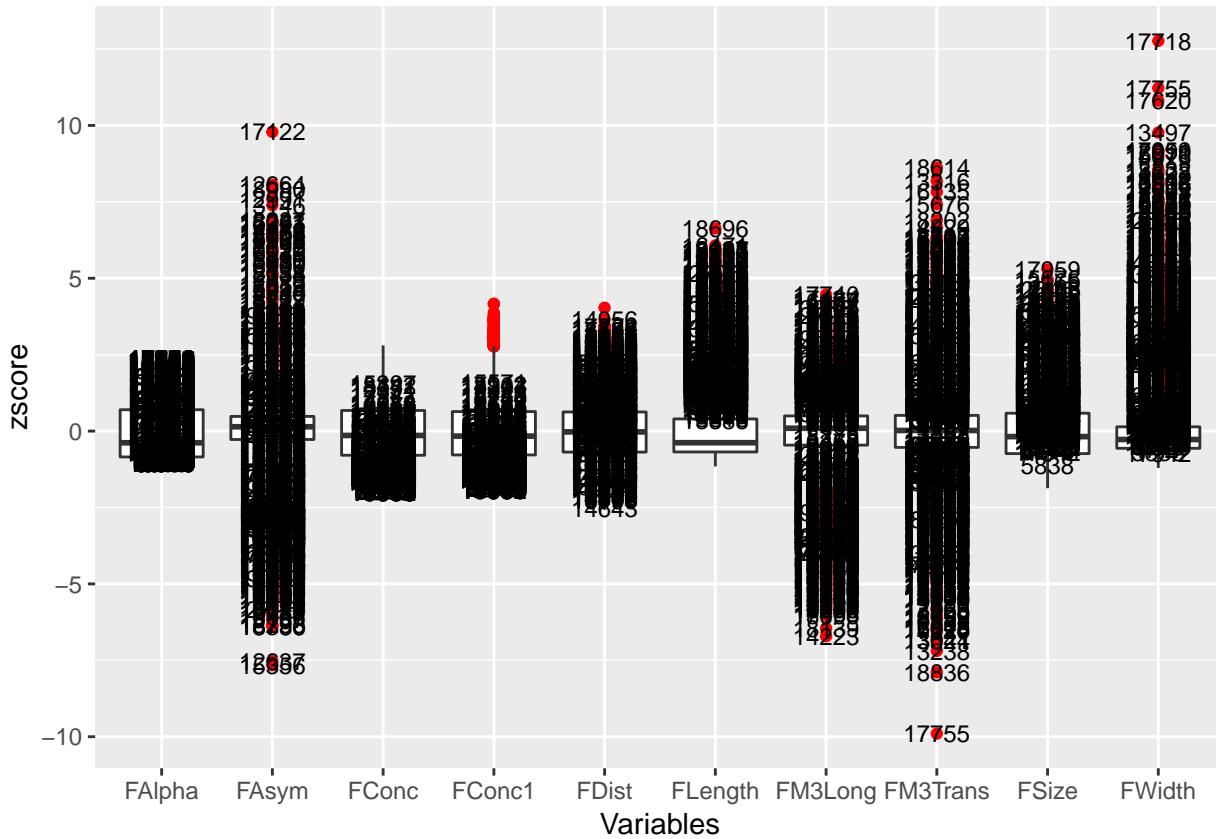
```
# BoxPlot para outliers normales
MiBoxPlot_juntos(mydata.numeric)
```



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

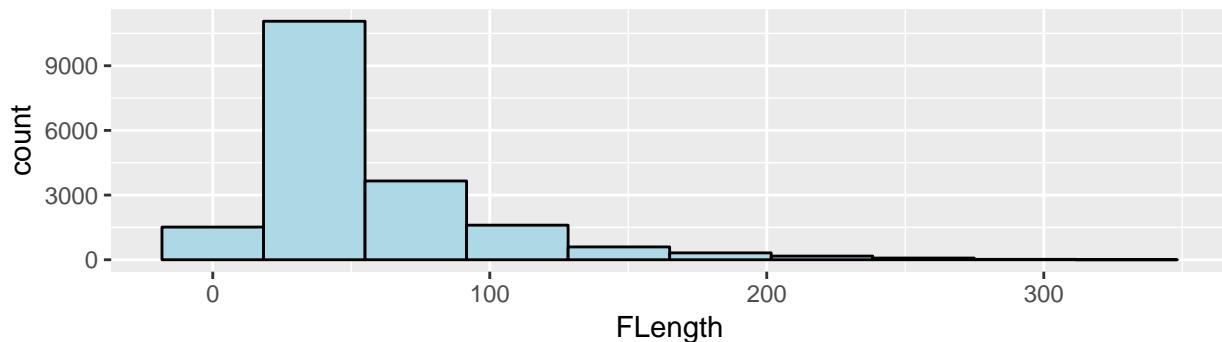
En este apartado realizaremos la detección de outliers en nuestro dataset con test estadísticos, como por ejemplo el test de Grubbs.

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

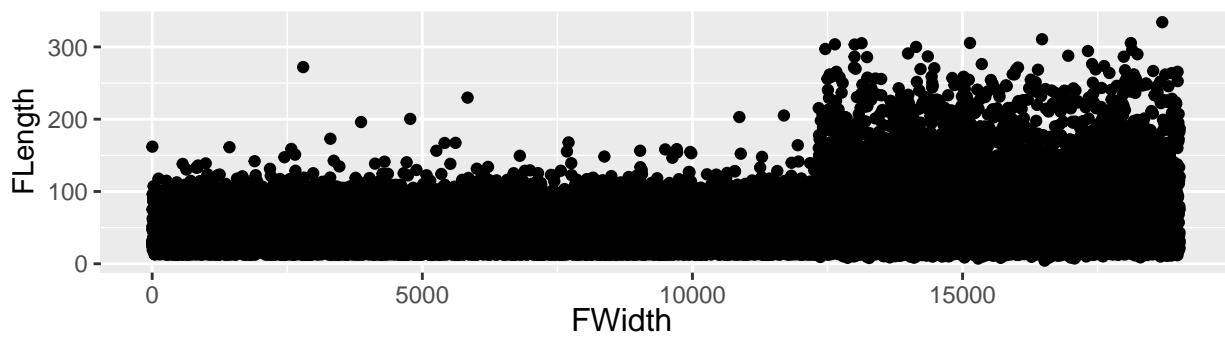
```
histogram_by = function(datos,var, bins=5){  
  
  ggplot(datos,aes_string(x=var)) +  
    geom_histogram(fill='lightblue', color="black", bins=bins)+  
    ggtitle("histogram")  
}  
  
scatter_by = function(datos,var){  
  x= seq_along(datos[,var])  
  ggplot(datos,aes_string(y=var,x=x))+  
    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

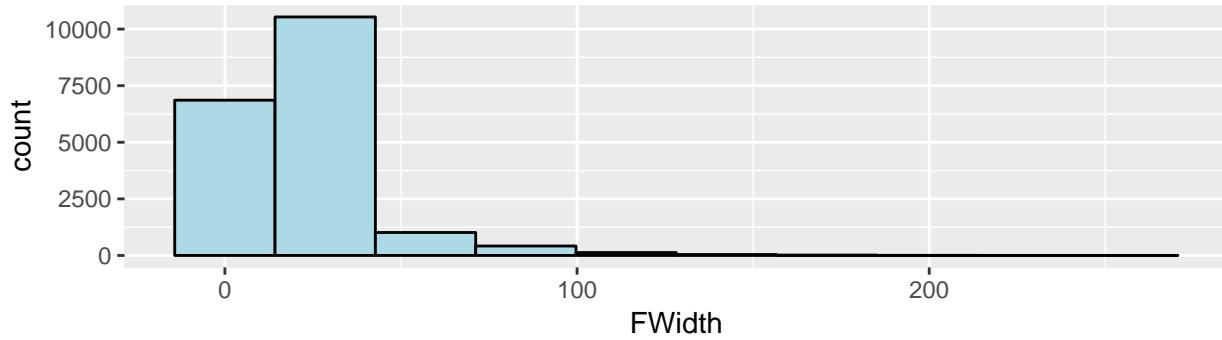
histogram



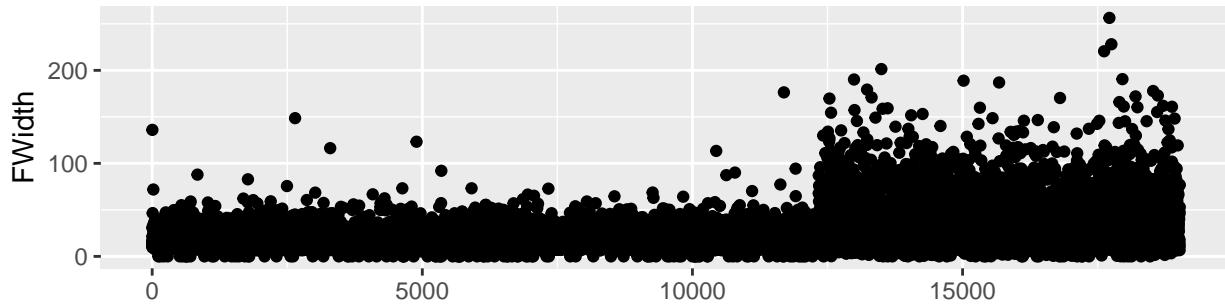
scatter



histogram

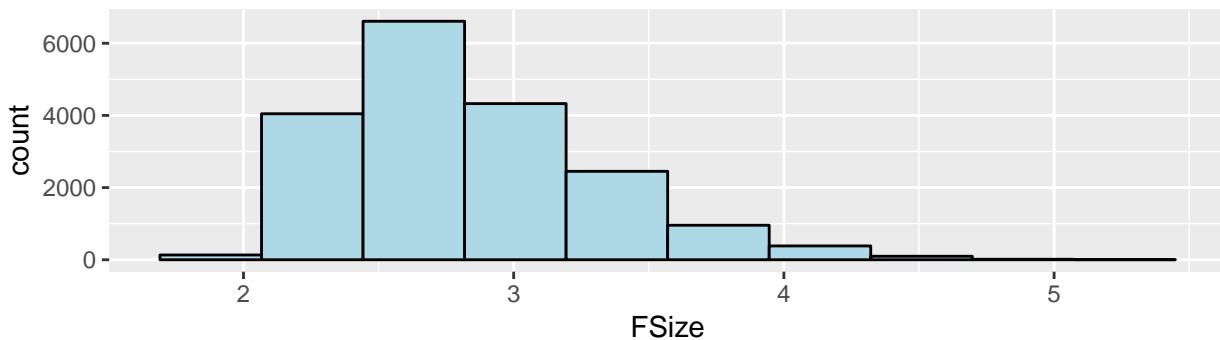


scatter

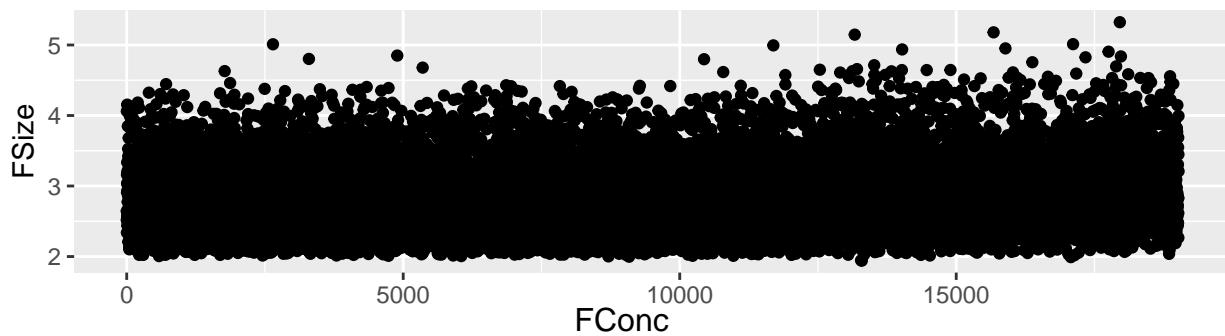


FSize

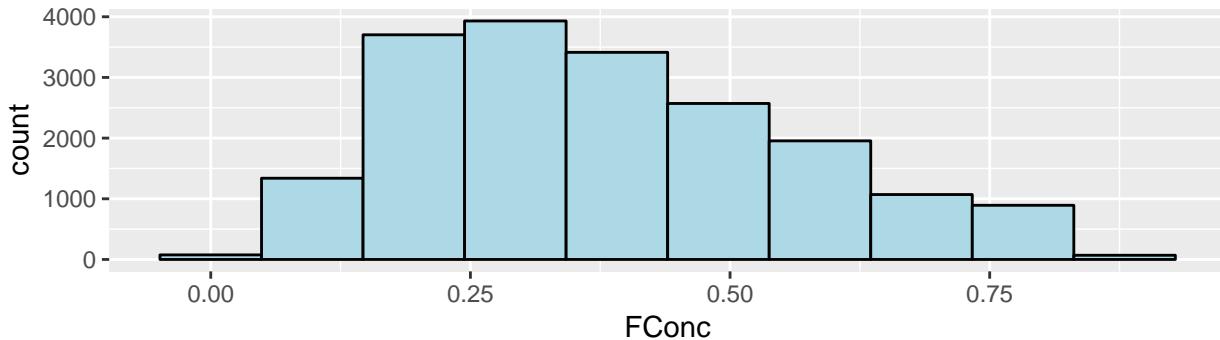
histogram



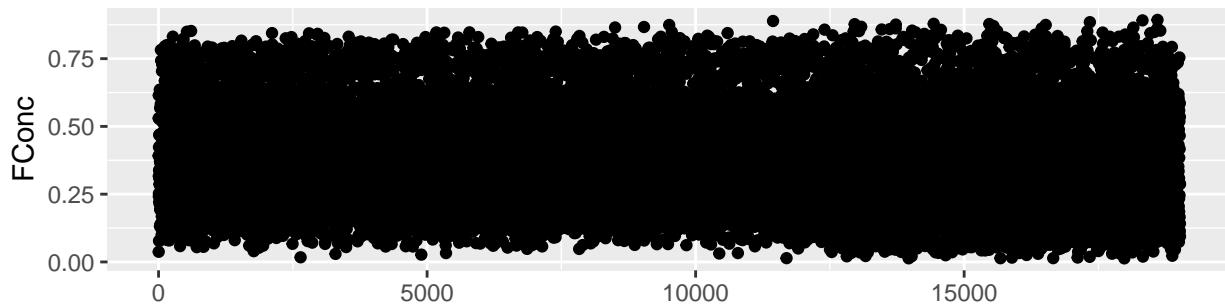
scatter



histogram

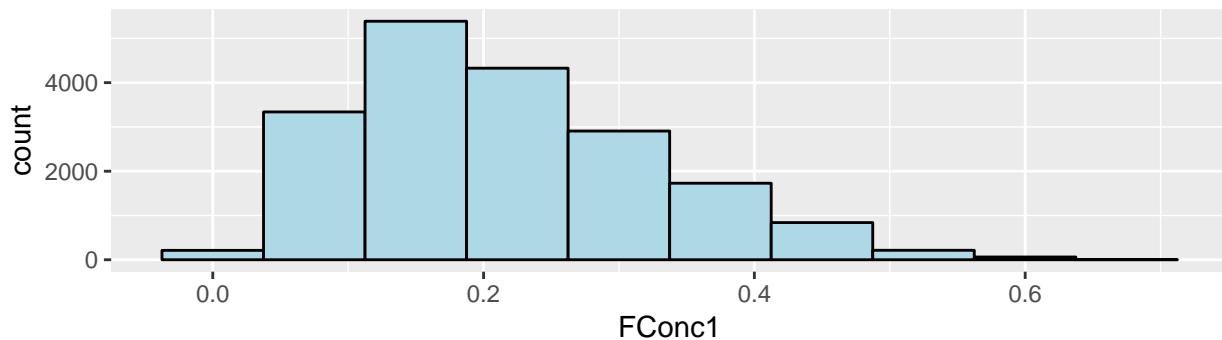


scatter

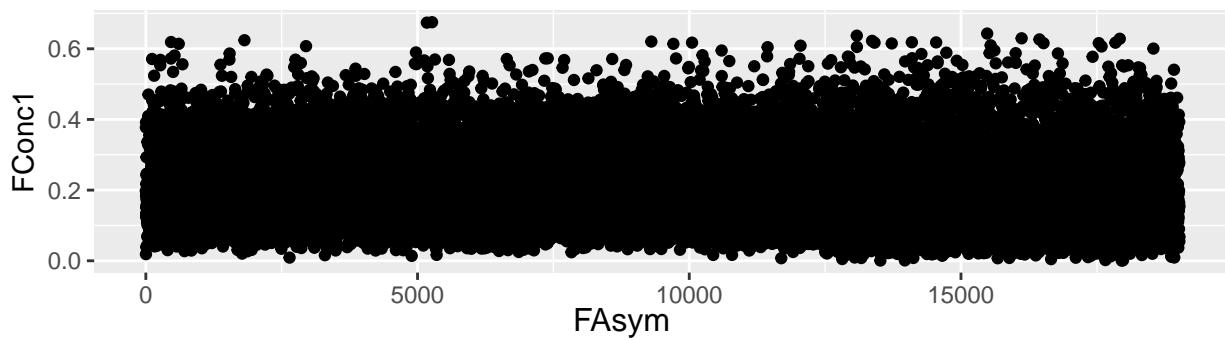


FConc1

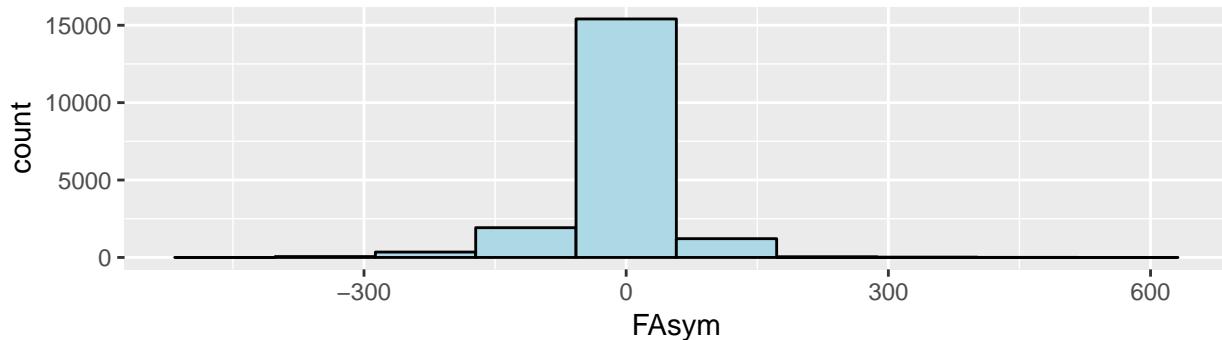
histogram



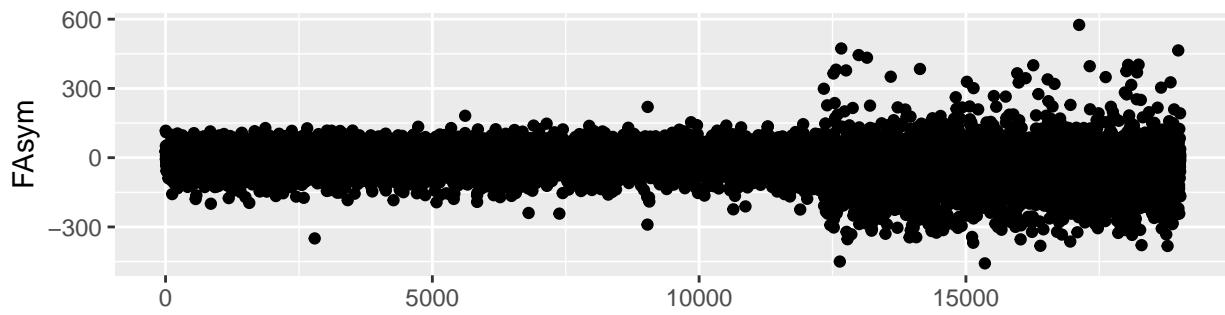
scatter



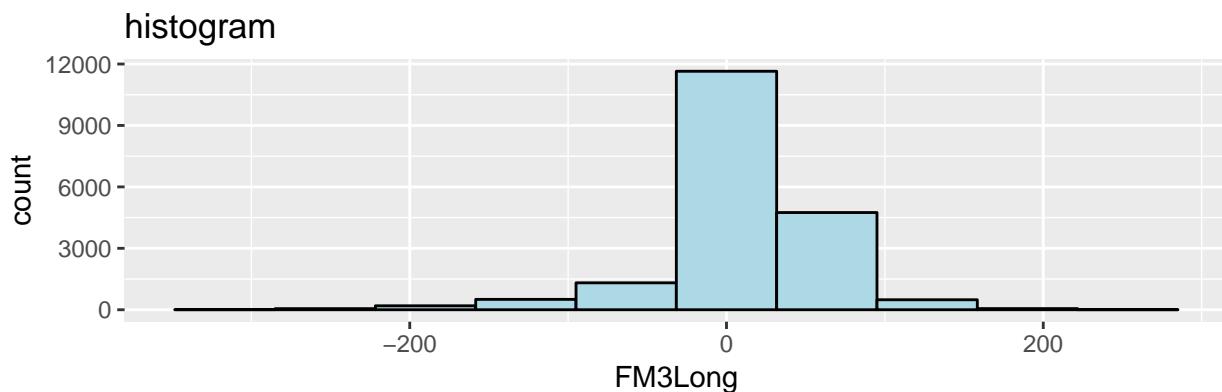
histogram



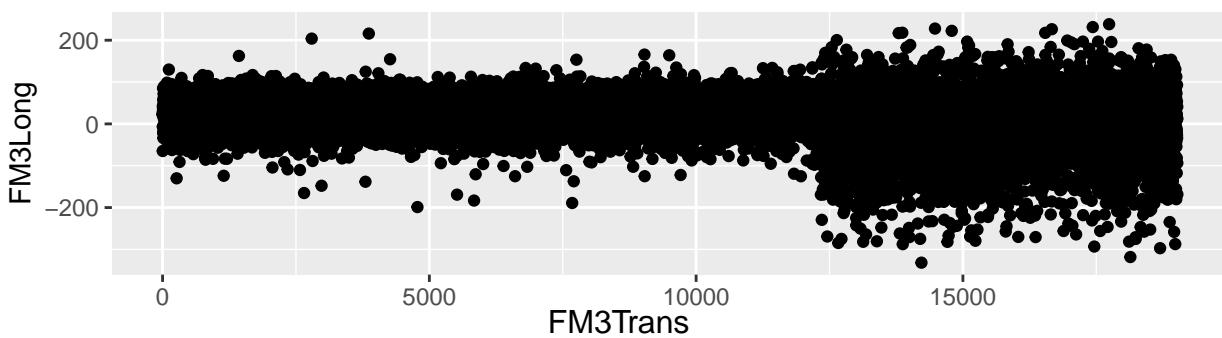
scatter



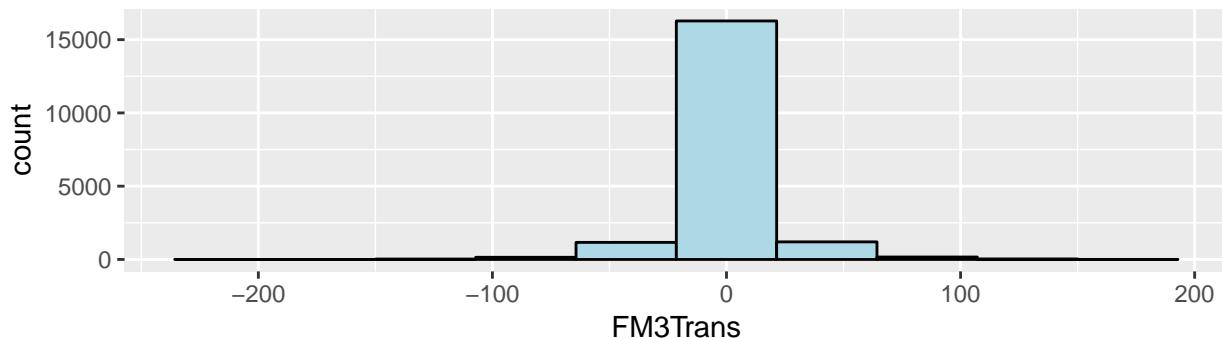
FM3Long



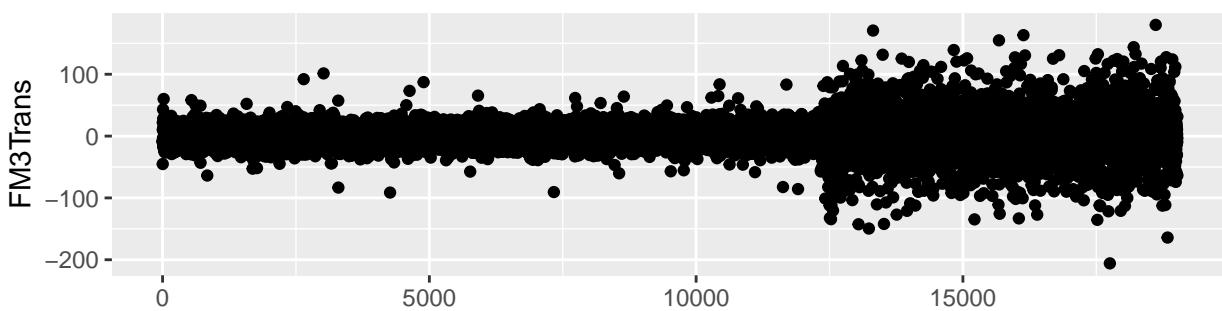
scatter



histogram

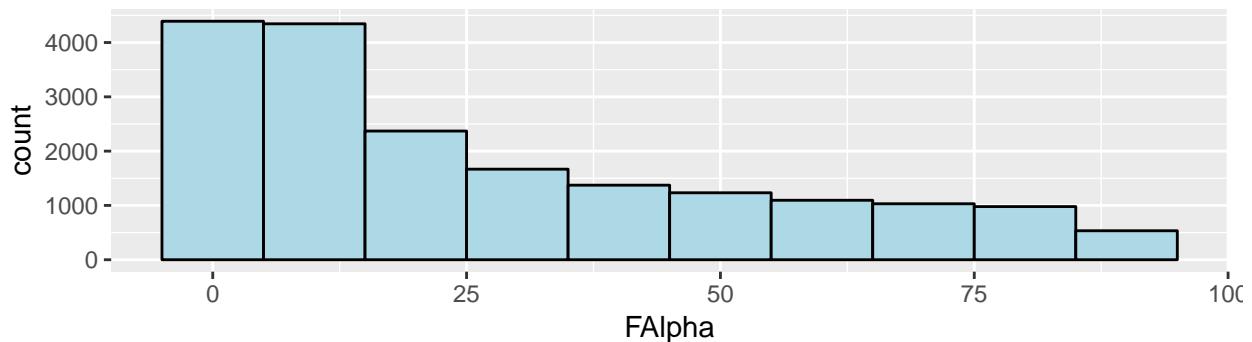


scatter

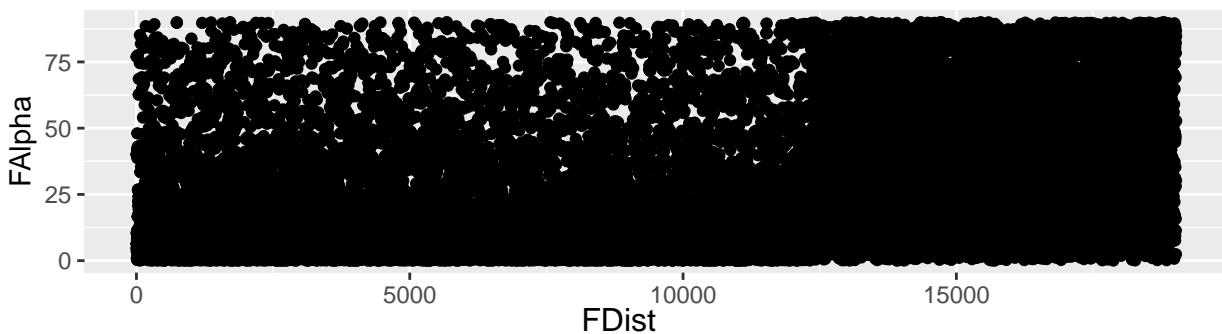


FAlpha

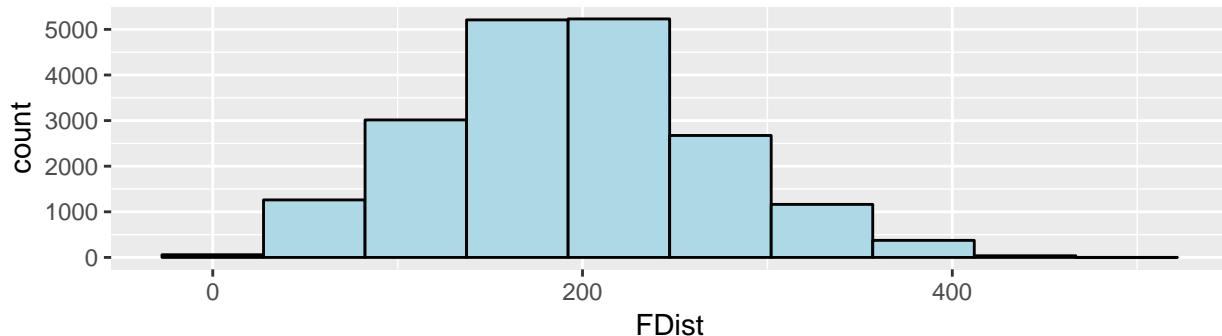
histogram



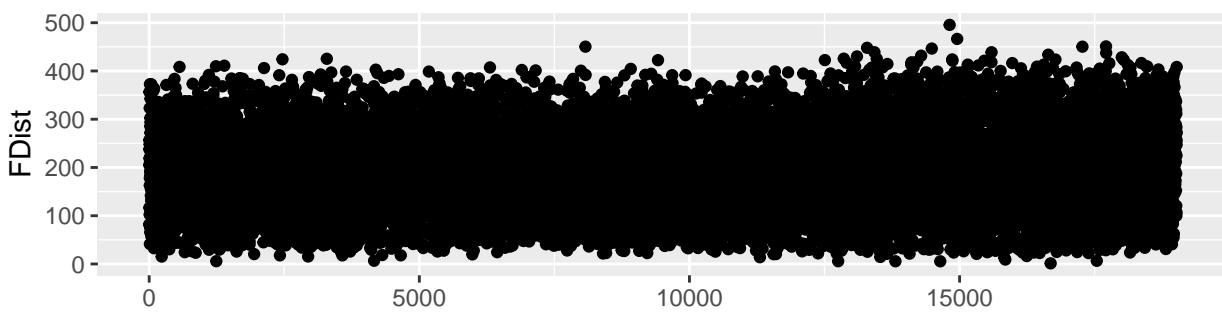
scatter



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

```

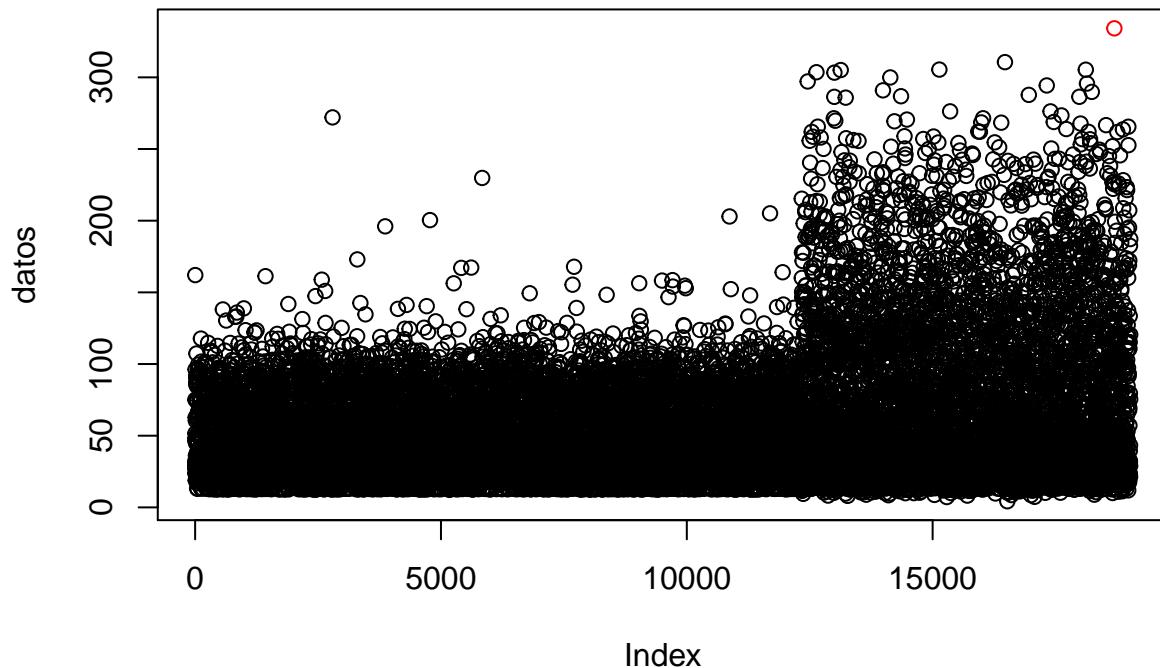
Ahora que ya tenemos los indices de los outliers, podemos mostrar un gráfico identificando los outliers de cada variable, para ello podemos utilizar la función *MiPlot_Univariate_Outliers()* que ya viene implementada.

```

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

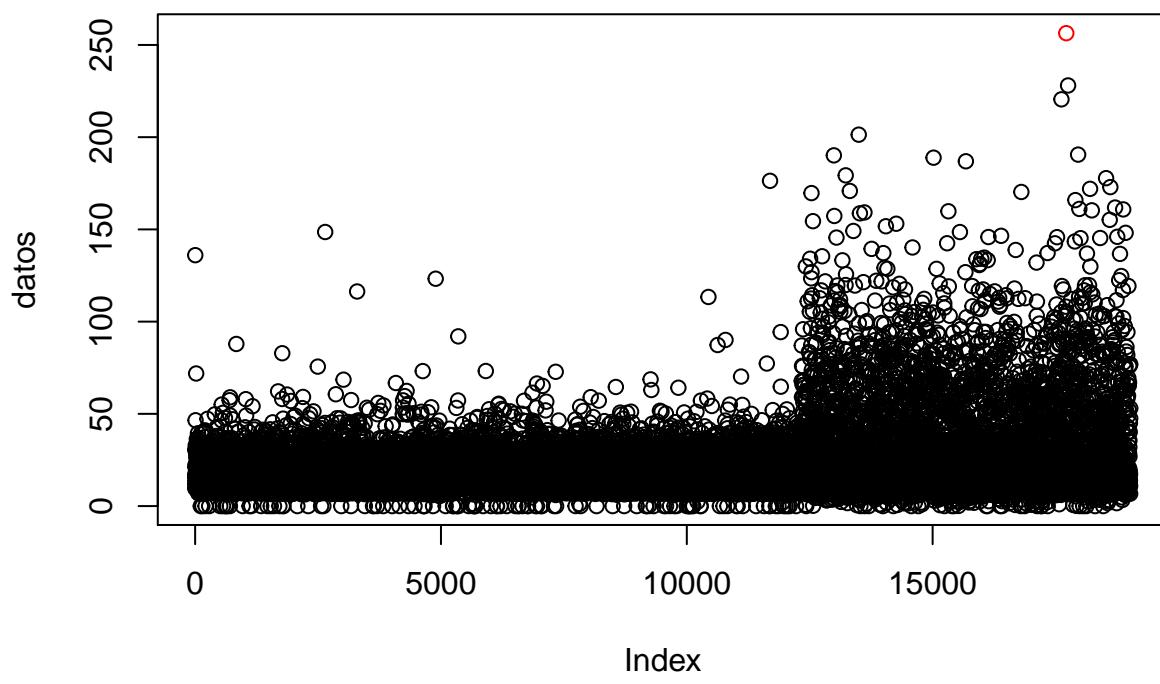
```

Outlier en los datos de FLength



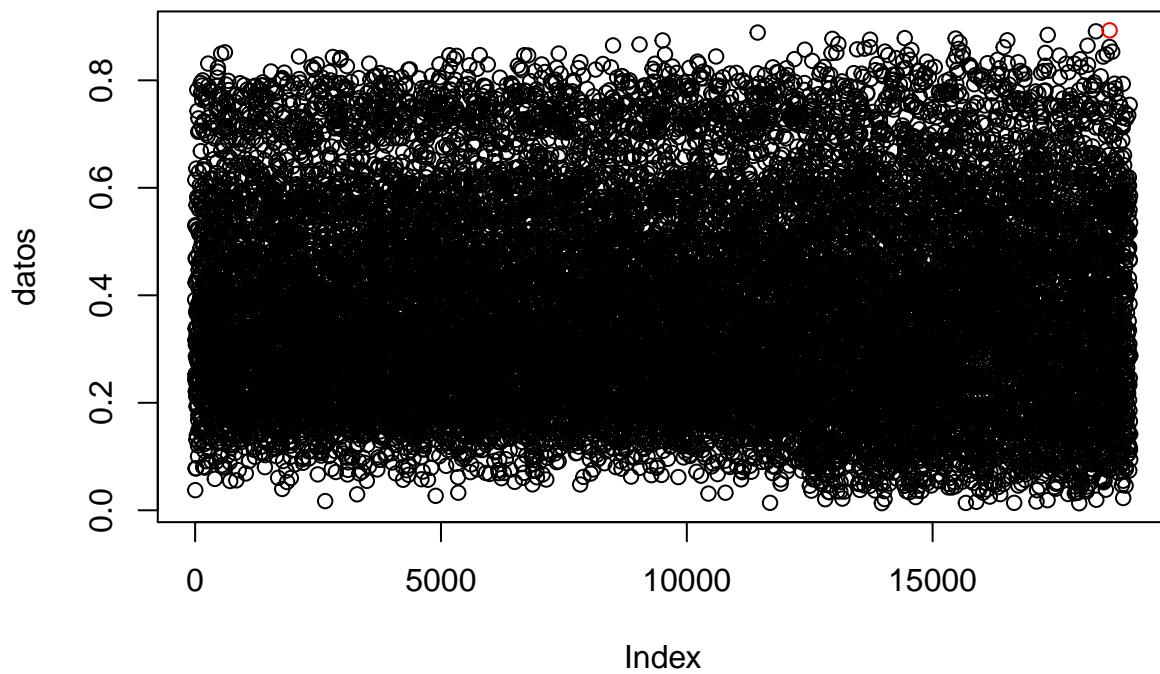
```
MiPlot_Univariate_Outliers(mydata.numeric$FWidth,indices.outliers[2],"Outlier en los datos de FWidth")
```

Outlier en los datos de FWidth



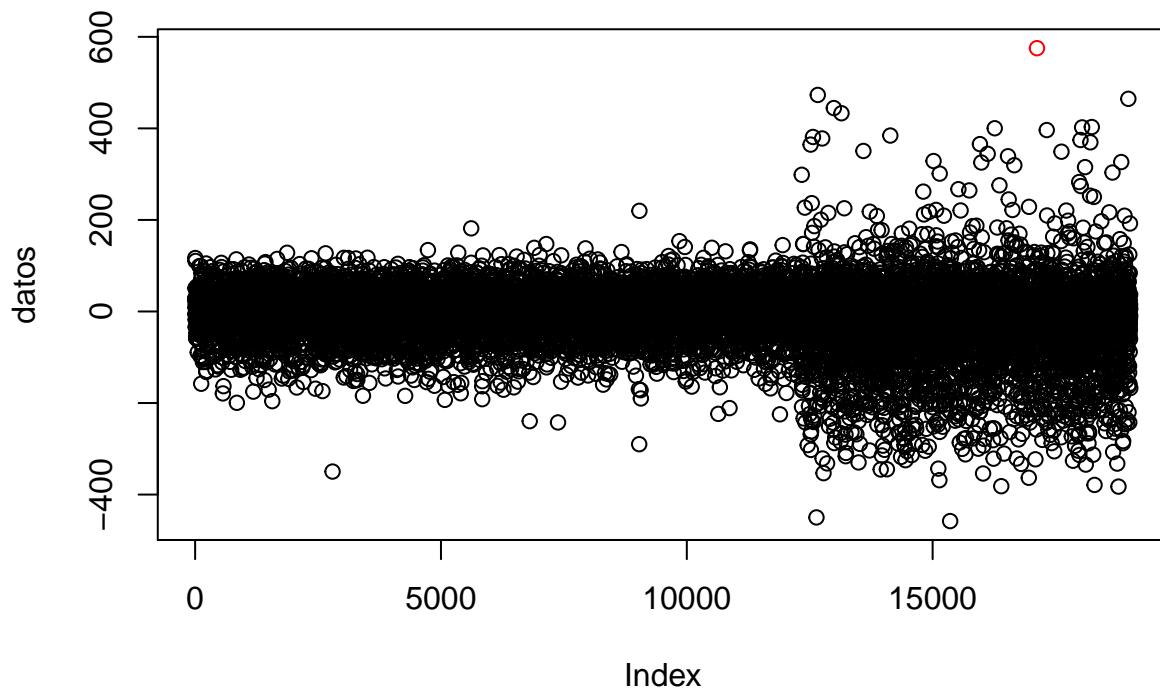
```
MiPlot_Univariate_Outliers(mydata.numeric$FConc,indices.outliers[3],"Outlier en los datos de FConc")
```

Outlier en los datos de FConc



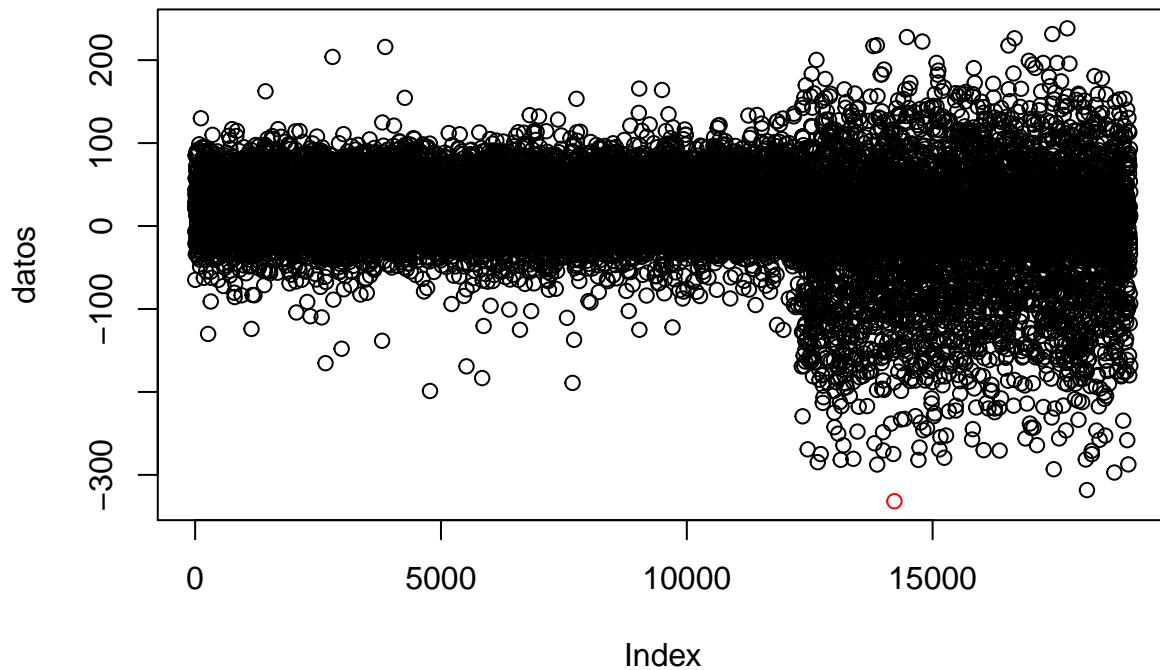
```
MiPlot_Univariate_Outliers(mydata.numeric$FAsym,indices.outliers[4],"Outlier en los datos de FAsym")
```

Outlier en los datos de FAsym



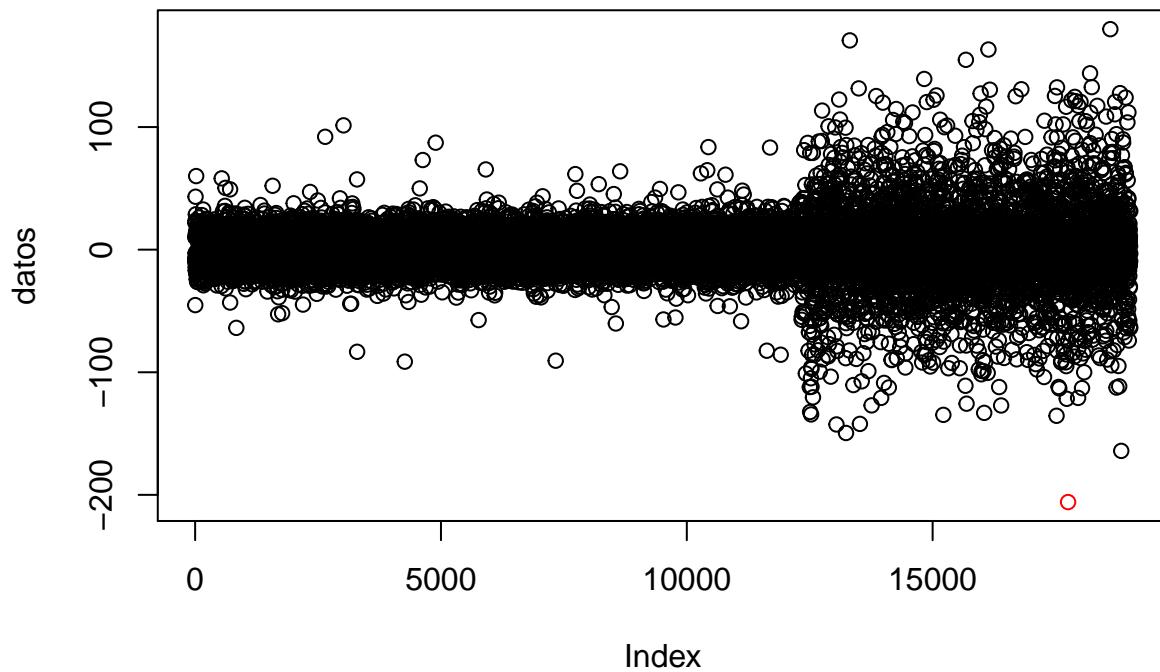
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Long,indices.outliers[5],"Outlier en los datos de FM3Long")
```

Outlier en los datos de FM3Long



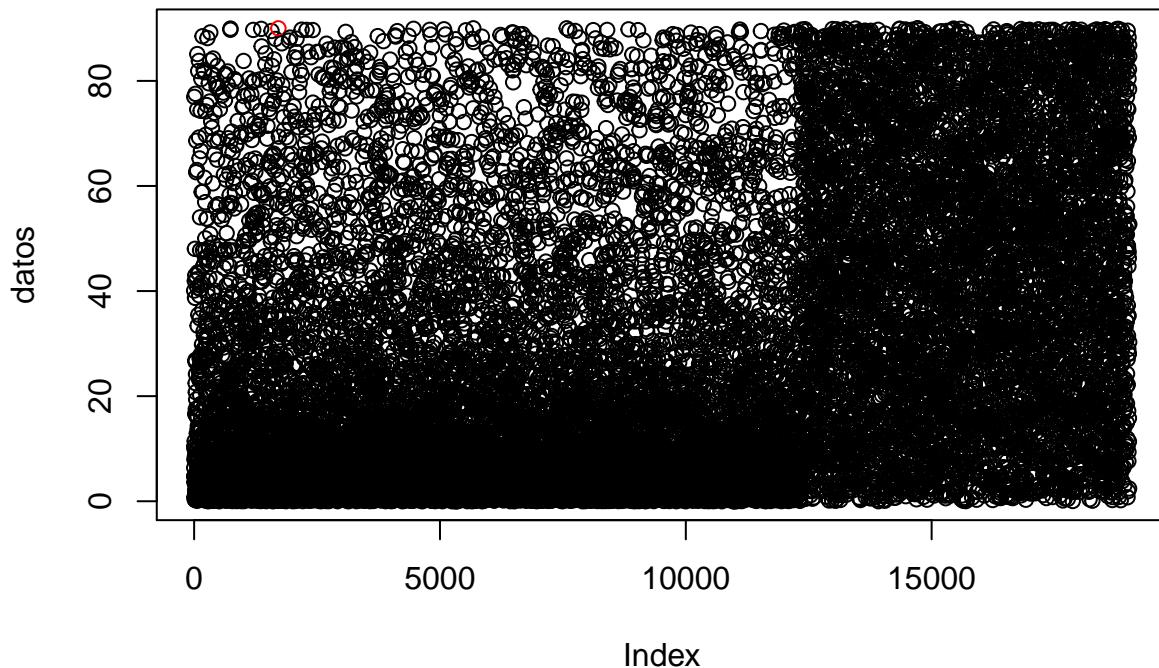
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Trans,indices.outliers[6],"Outlier en los datos de FM3Trans")
```

Outlier en los datos de FM3Trans



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

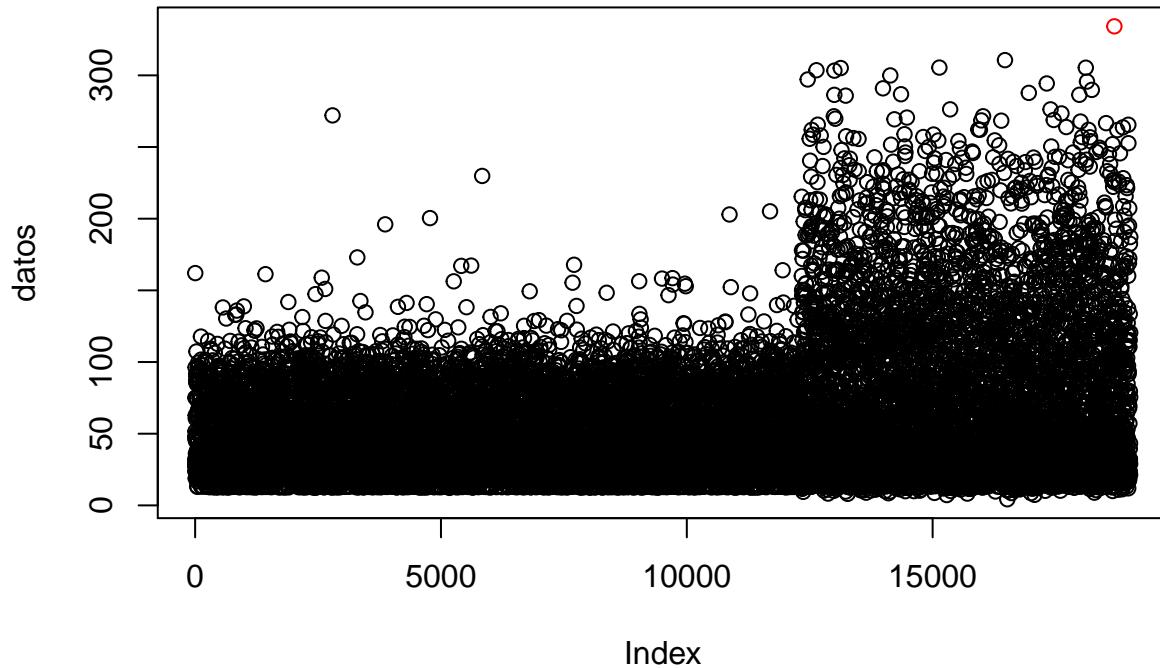
Outlier en los datos de FAlpha



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.

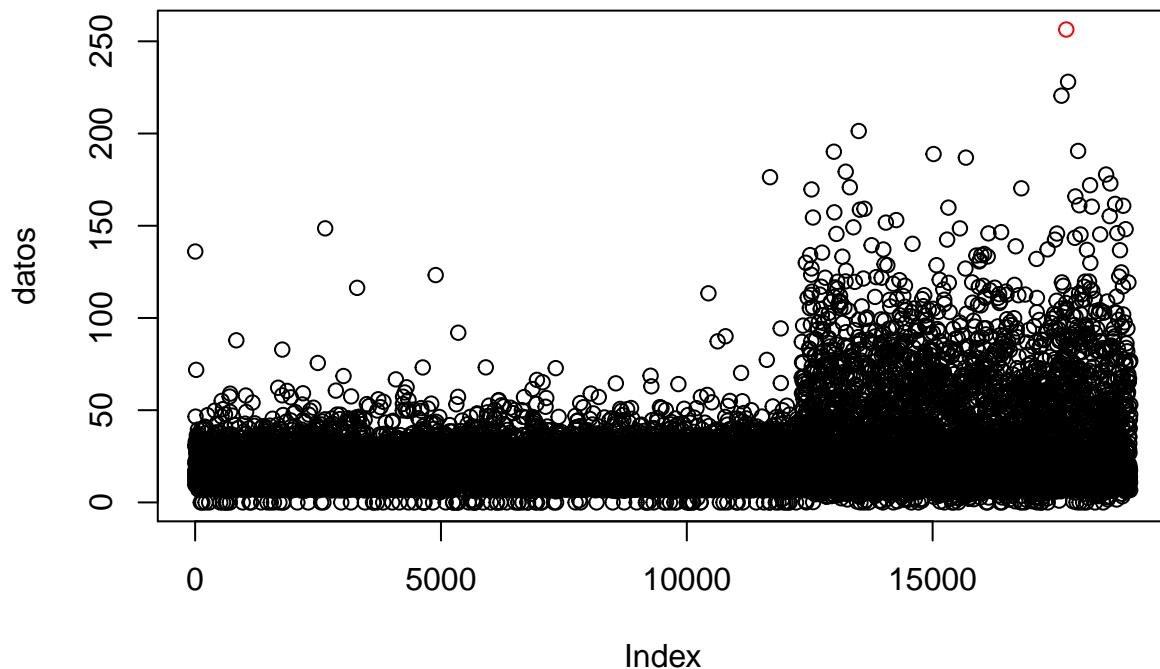
```
aux = mydata.numeric  
apply(aux, 2, MiPlot_resultados_TestGrubbs)  
  
## p.value: 6.177066e-07  
## ?ndice de outlier: 18696  
## Valor del outlier: 334.177
```

Test de Grubbs



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

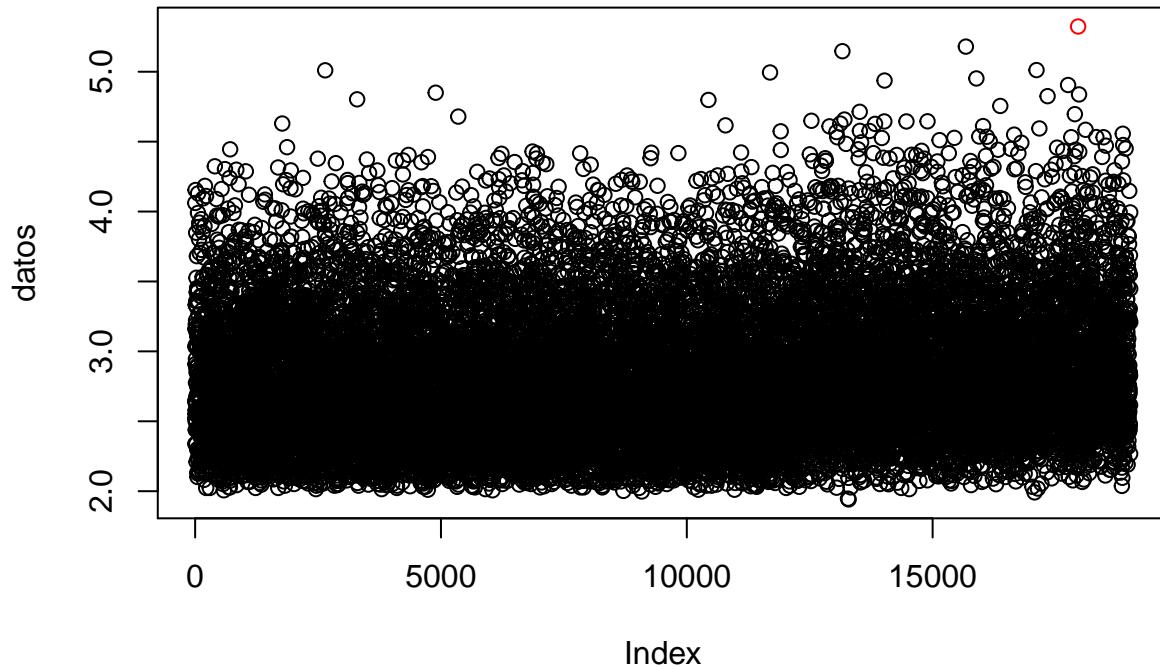
Test de Grubbs



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

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

Test de Grubbs

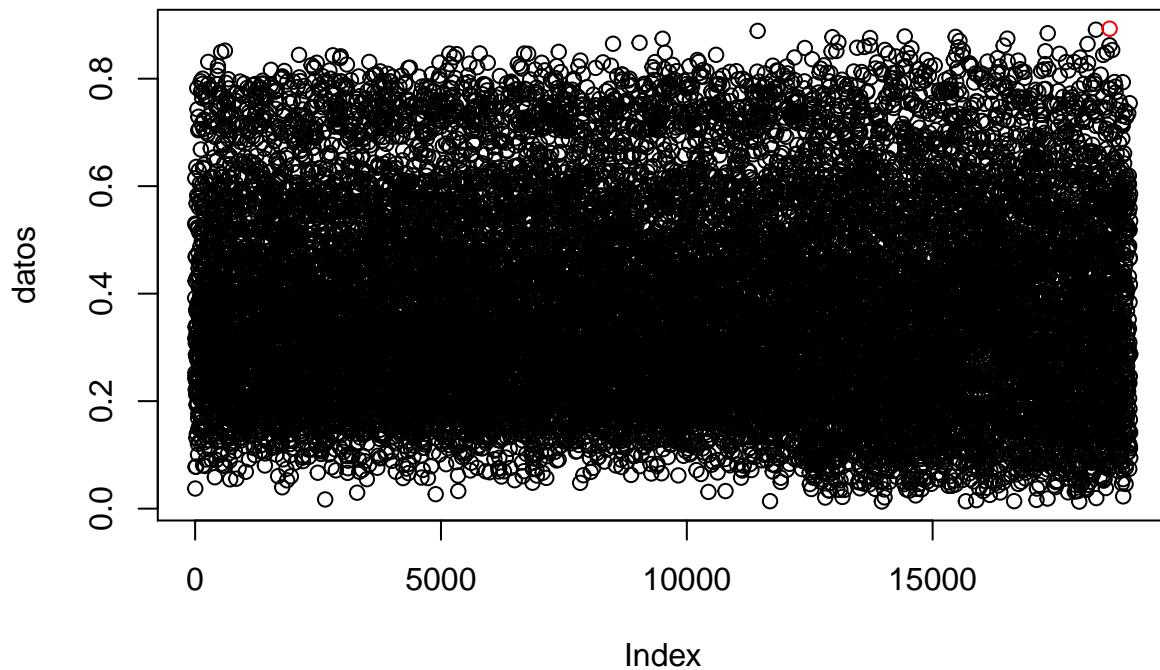


```
## p.value: 0
```

```
## ?ndice de outlier: 18600
```

```
## Valor del outlier: 0.893
```

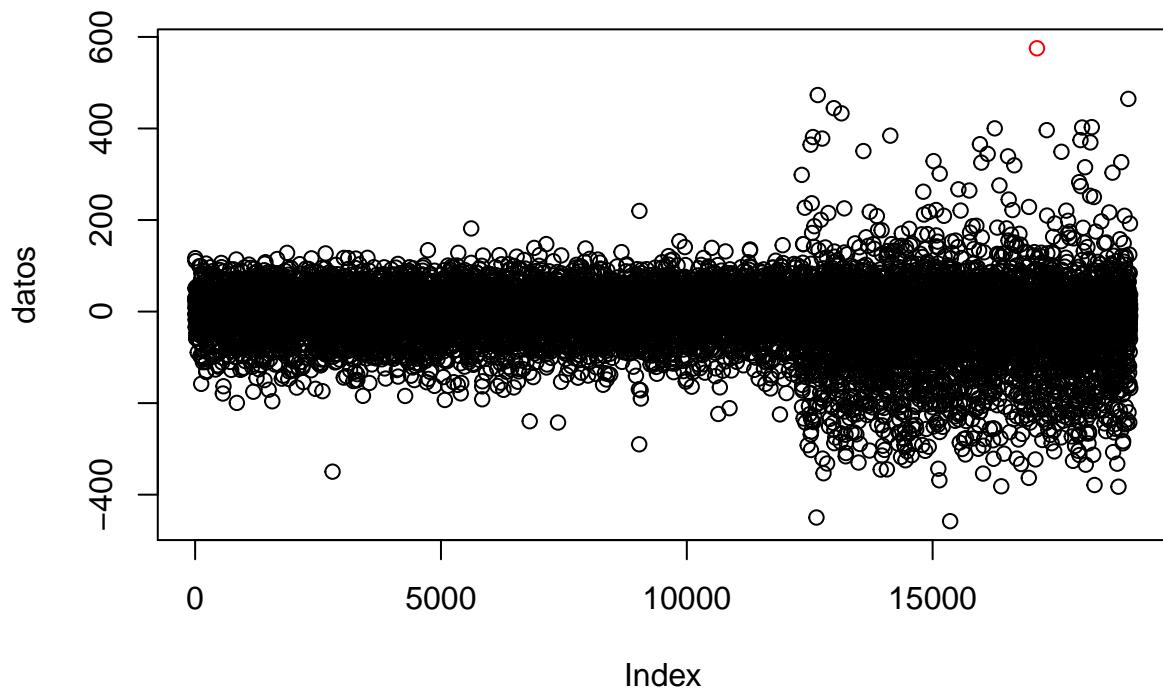
Test de Grubbs



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

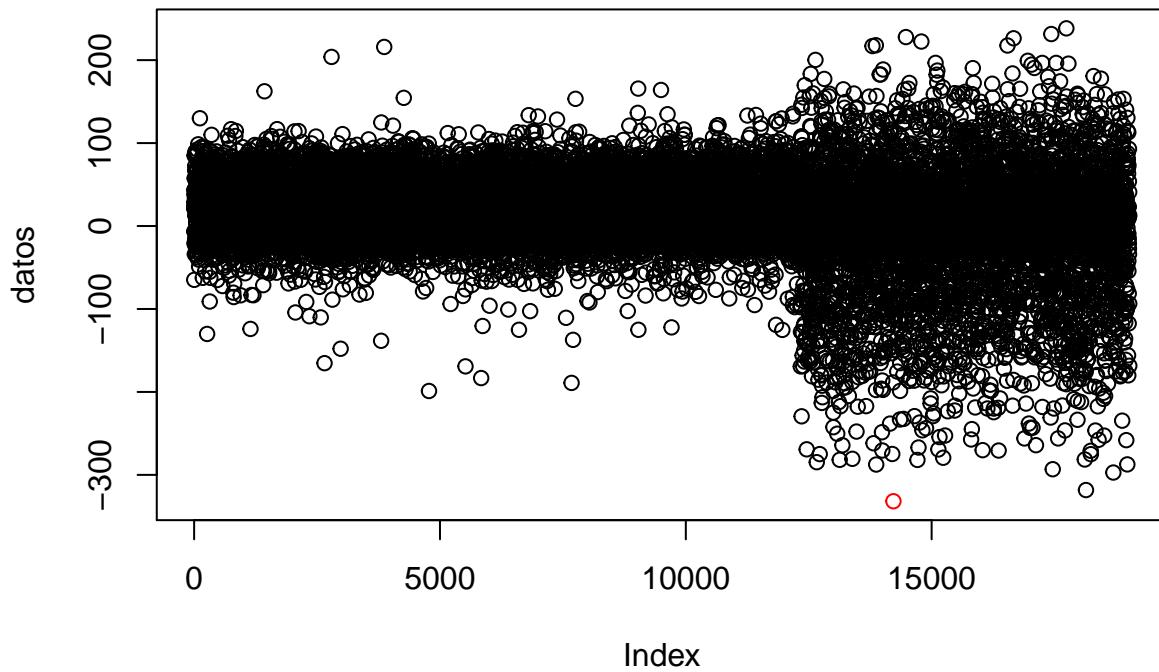
```
## No hay outliers p.value: 0  
## ?ndice de outlier: 17122  
## Valor del outlier: 575.2407
```

Test de Grubbs



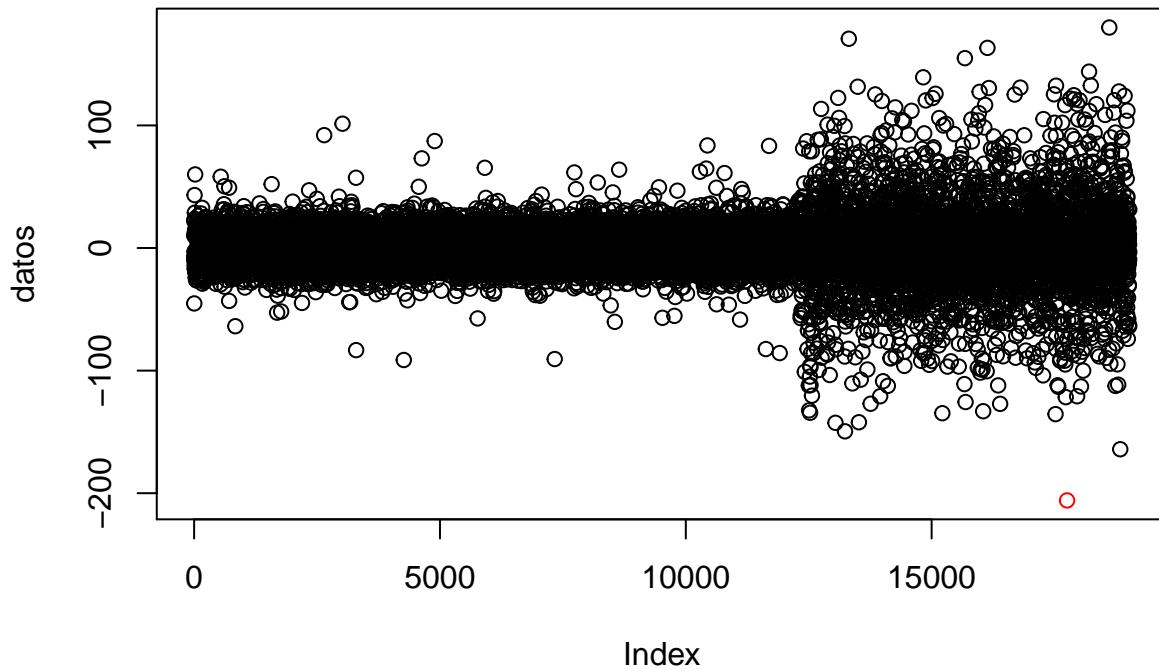
```
## p.value: 3.549083e-07  
## ?ndice de outlier: 14223  
## Valor del outlier: -331.78
```

Test de Grubbs



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

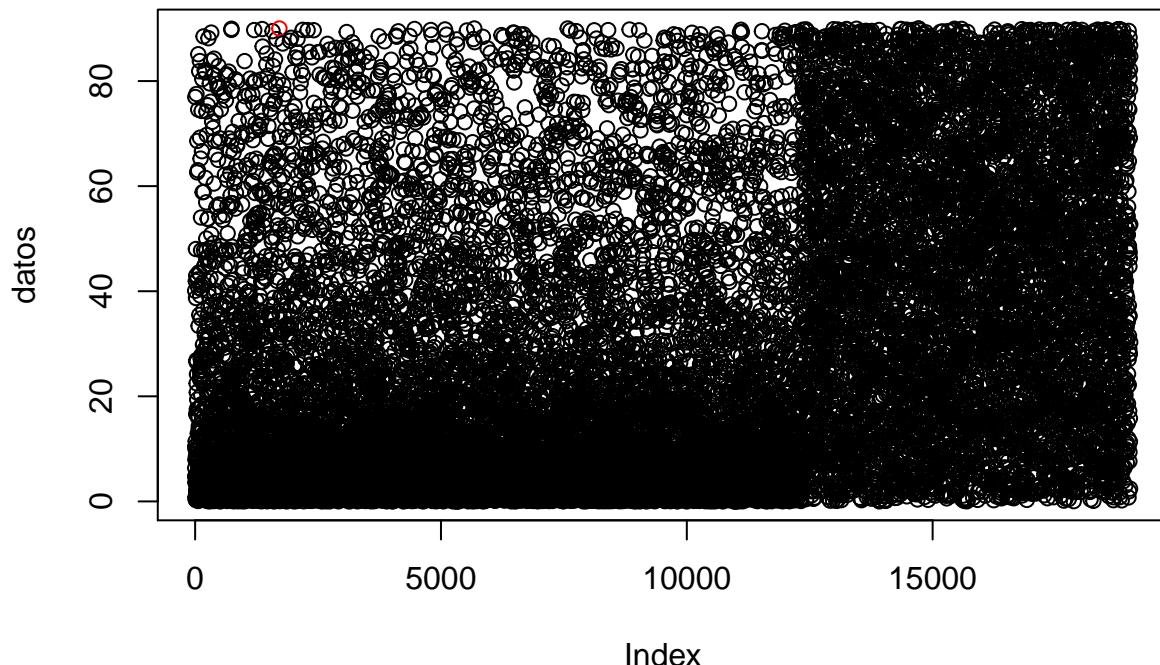
Test de Grubbs



```
## 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.

```

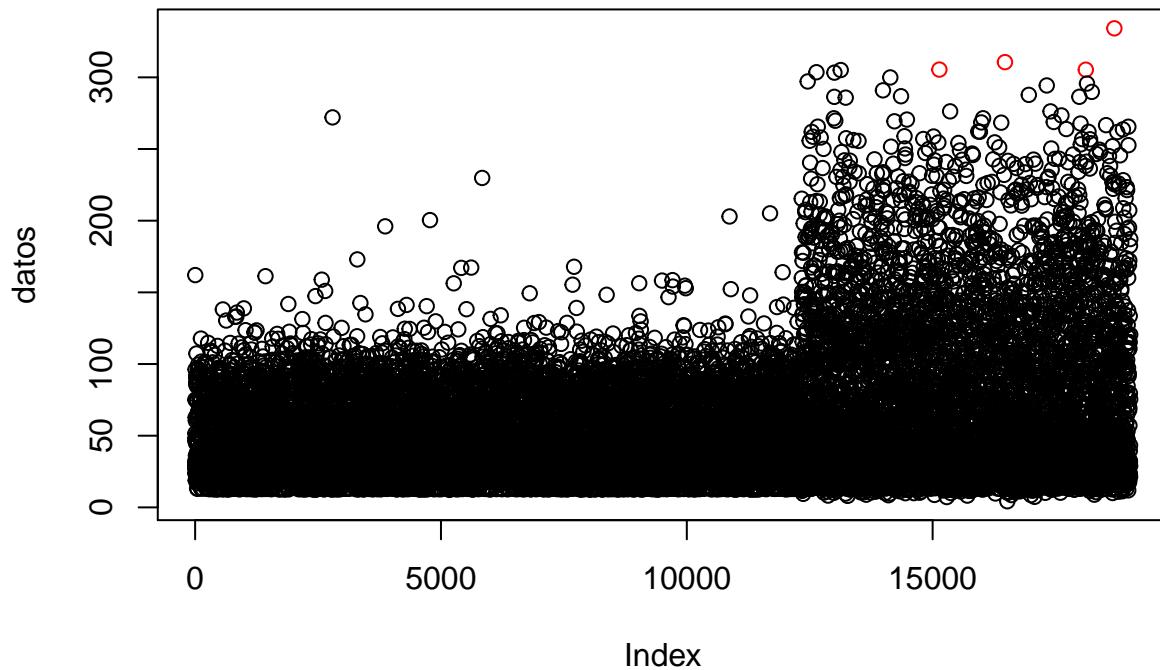
# Función para devolver la posición de los outliers reales (TRUE) del test de Rosner
obtener_outlier_reales_Rosner = function(data=list()){
  m_data = data.frame(bool=data$bools,ind=data$indices)
  resultados = as.vector(subset(m_data,bool==TRUE,select=ind)$ind)
  resultados
}

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")

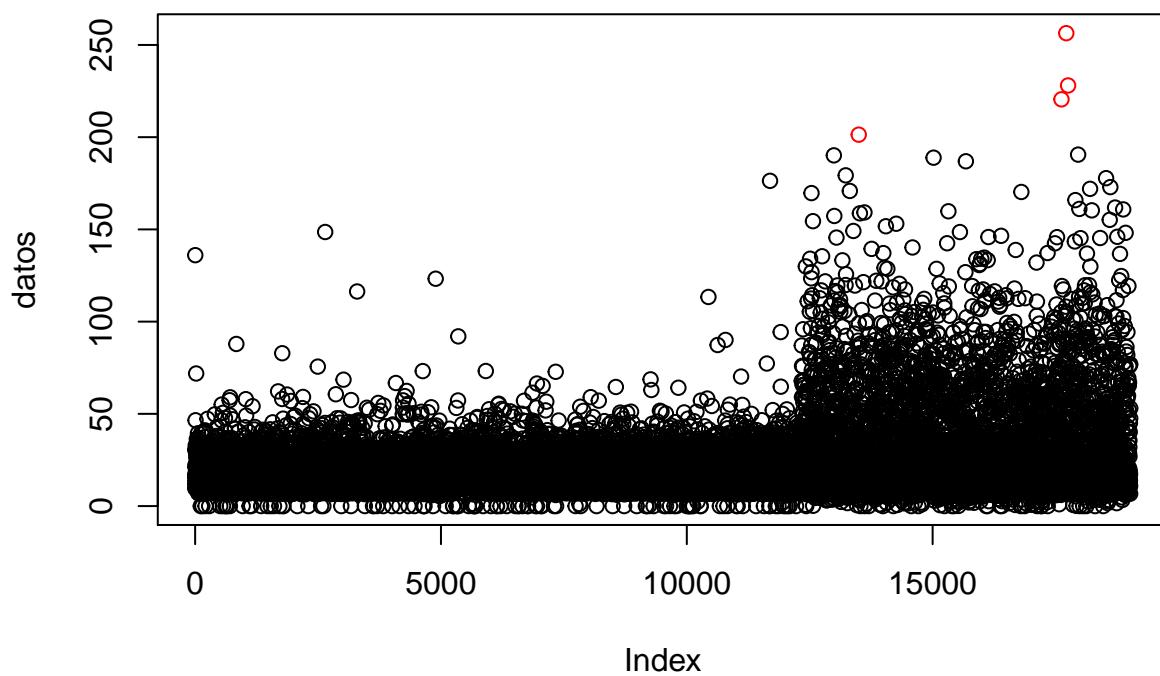
```

Outliers Rosner FLength



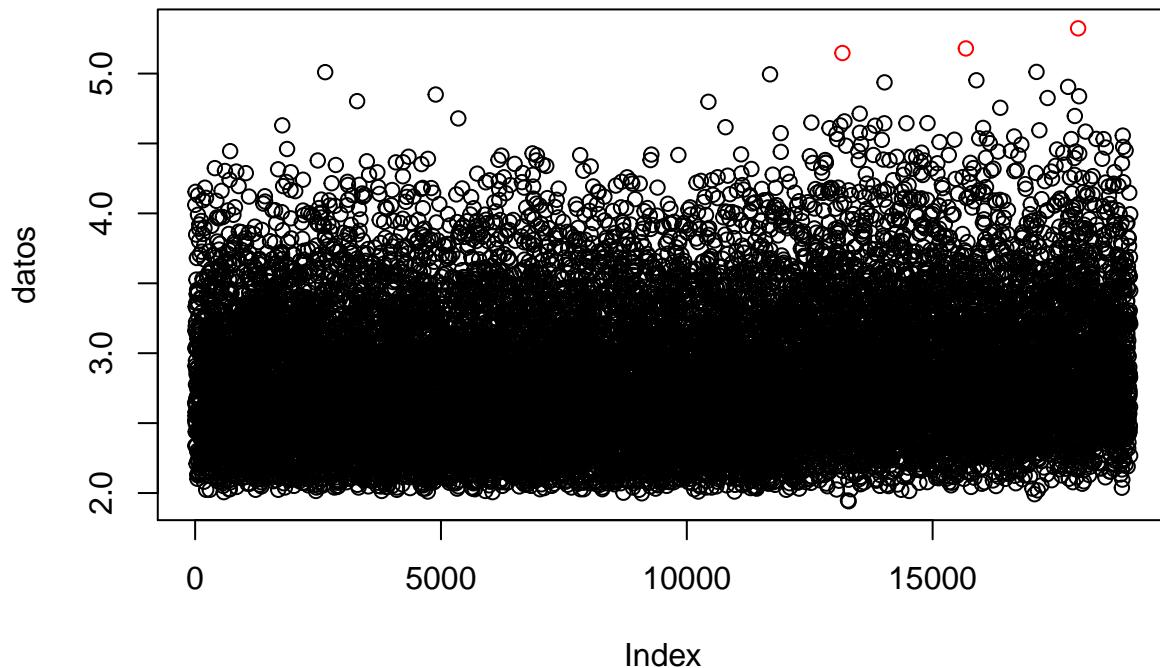
```
MiPlot_Univariate_Outliers(mydata.numeric$FWidth, indices.rosner$FWidth, "Outliers Rosner FWidth")
```

Outliers Rosner FWidth



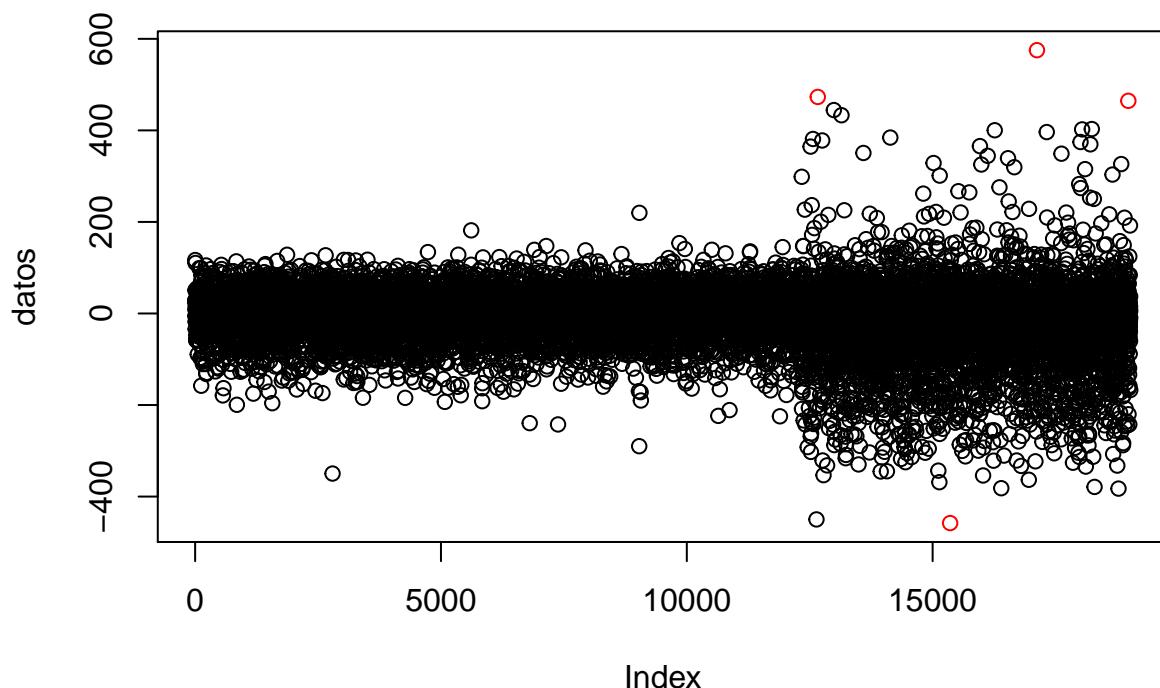
```
MiPlot_Univariate_Outliers(mydata.numeric$FSize, indices.rosner$FSize, "Outliers Rosner FSize")
```

Outliers Rosner FSize



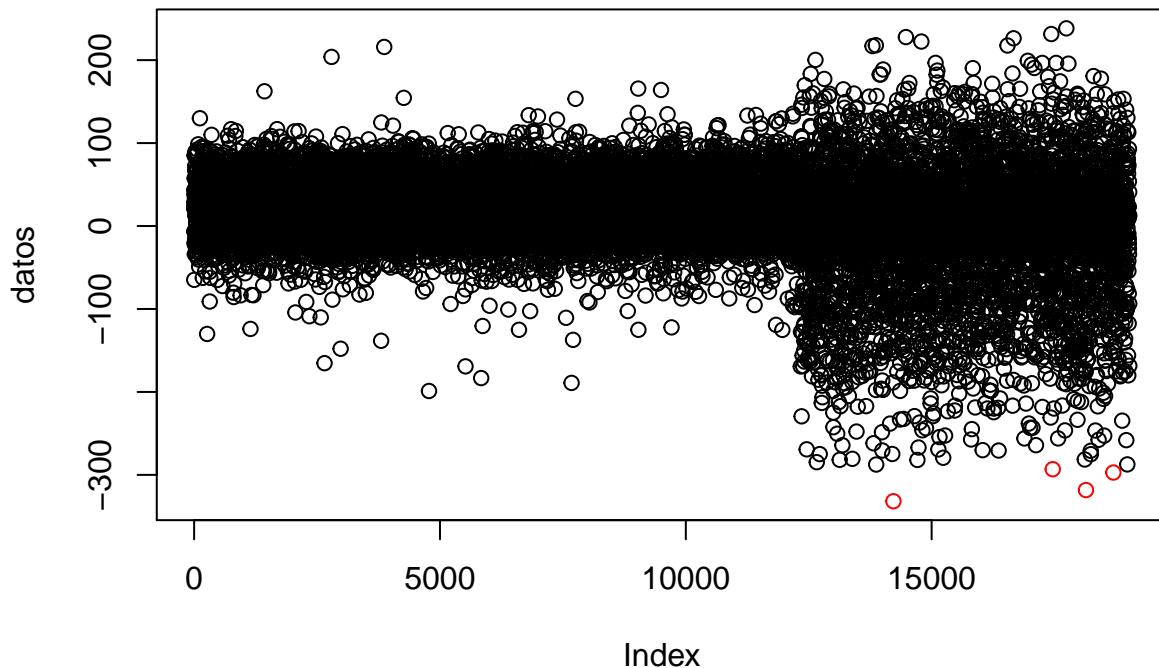
```
MiPlot_Univariate_Outliers(mydata.numeric$FAsym, indices.rosner$FAsym, "Outliers Rosner FAsym")
```

Outliers Rosner FAsym



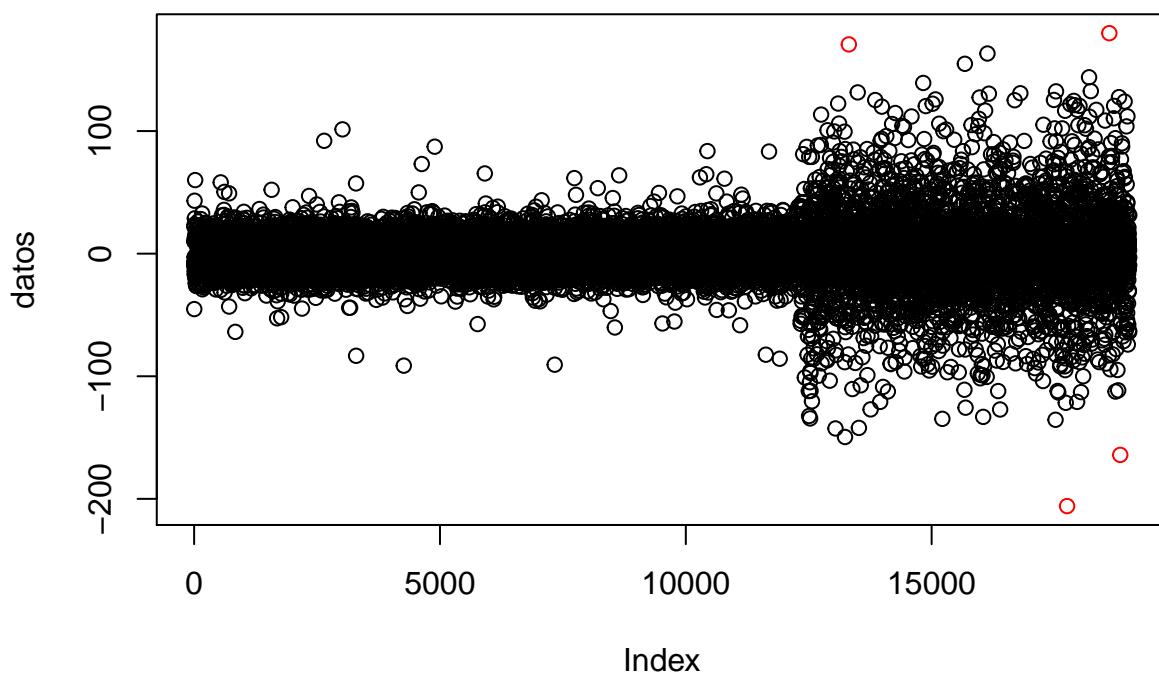
```
MiPlot_Univariate_Outliers(mydata.numeric$FM3Long, indices.rosner$FM3Long, "Outliers Rosner FM3Long")
```

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.

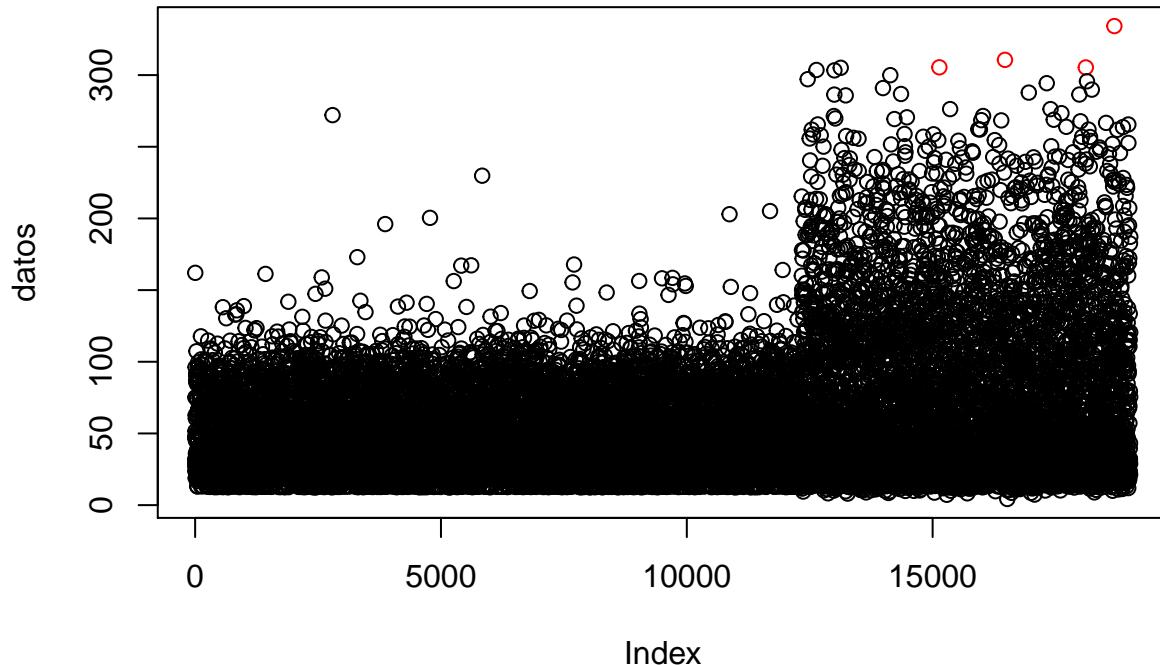
```

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

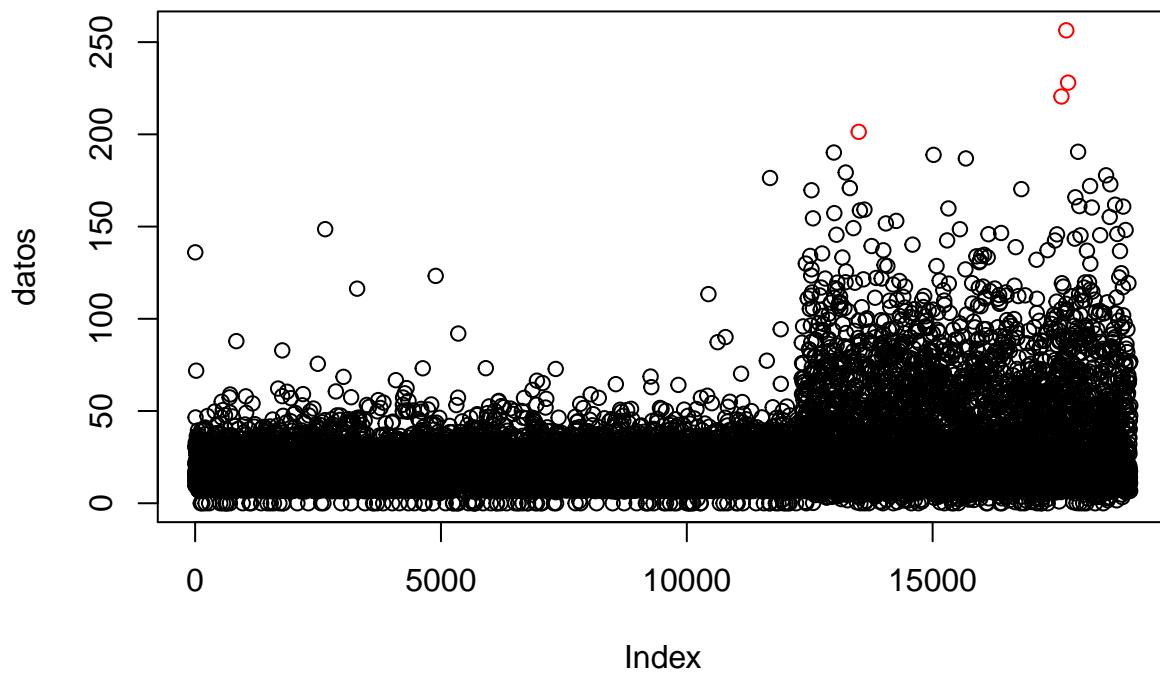


```

##
## 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

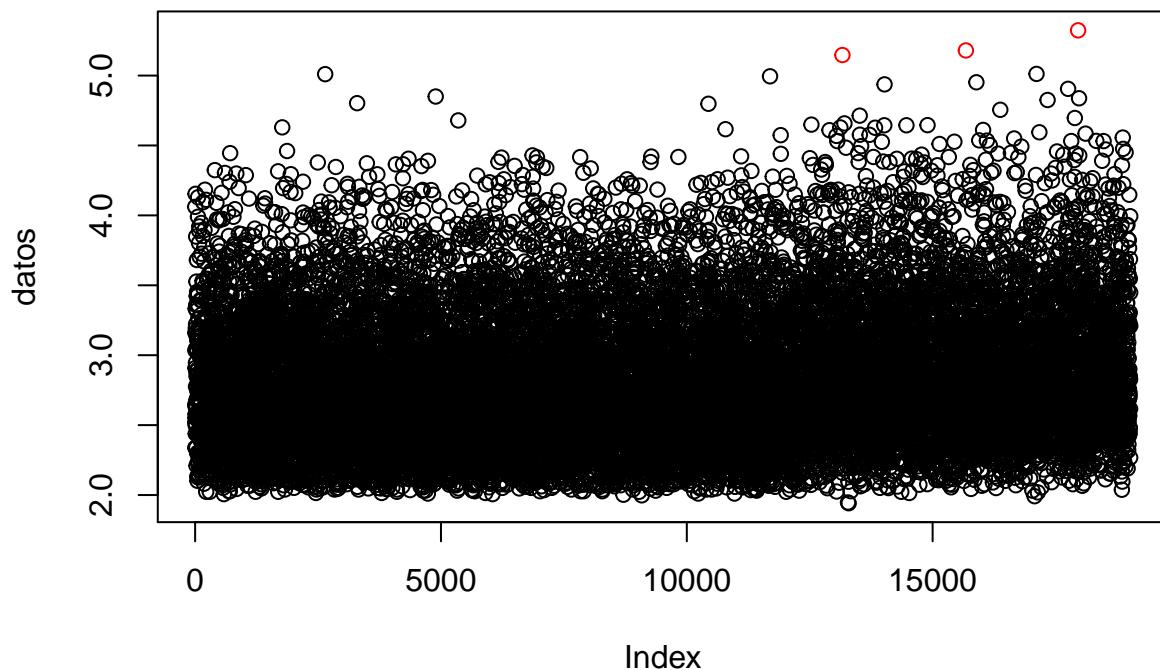
```

Test de Rosner



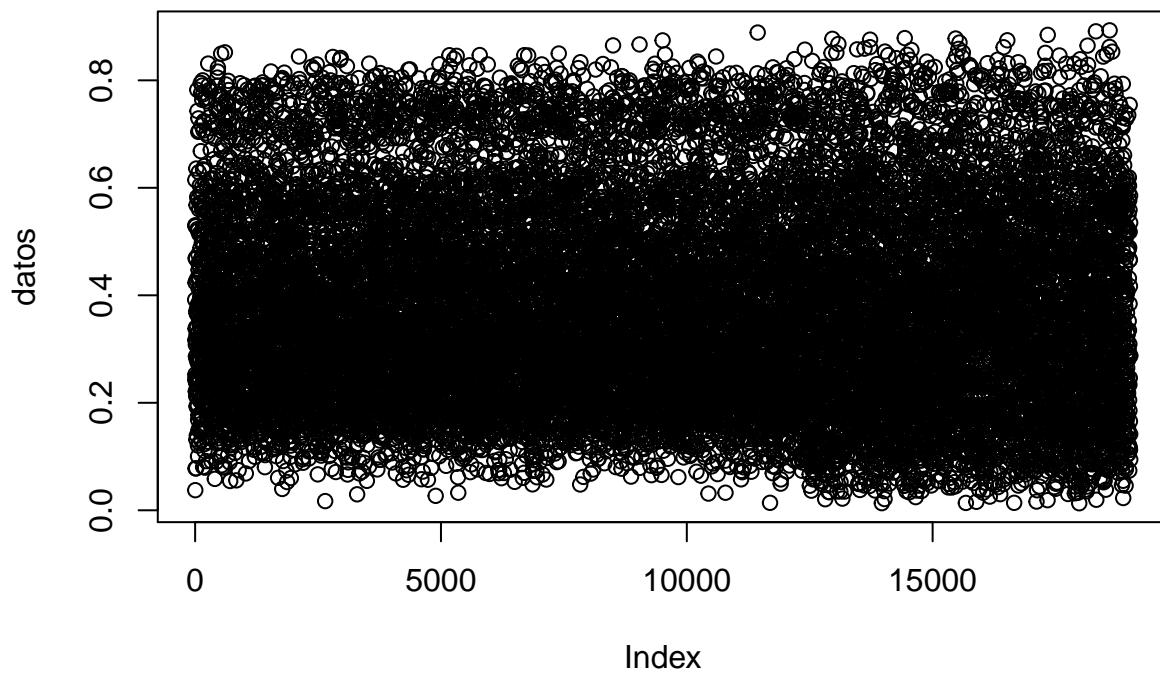
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17959 15676 13165 17113  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE FALSE  
## Los ?ndices de los outliers son: 17959 15676 13165  
## Los valores de los outliers son: 5.3233 5.1795 5.1467
```

Test de Rosner



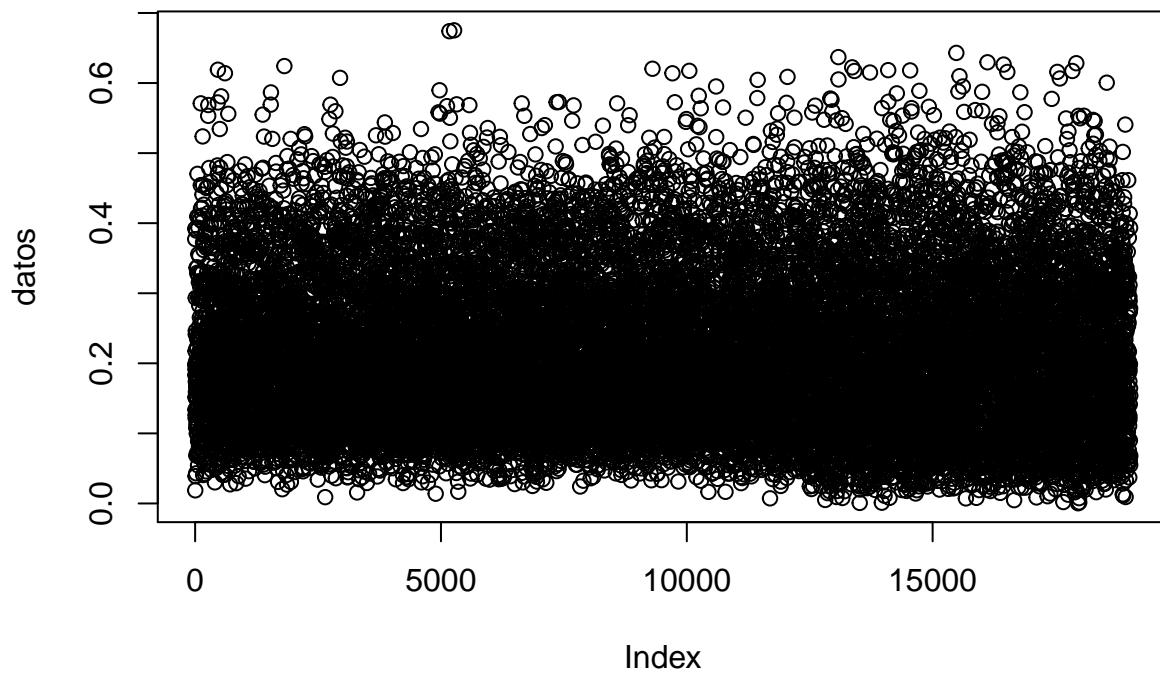
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 18600 18321 11442 17341  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



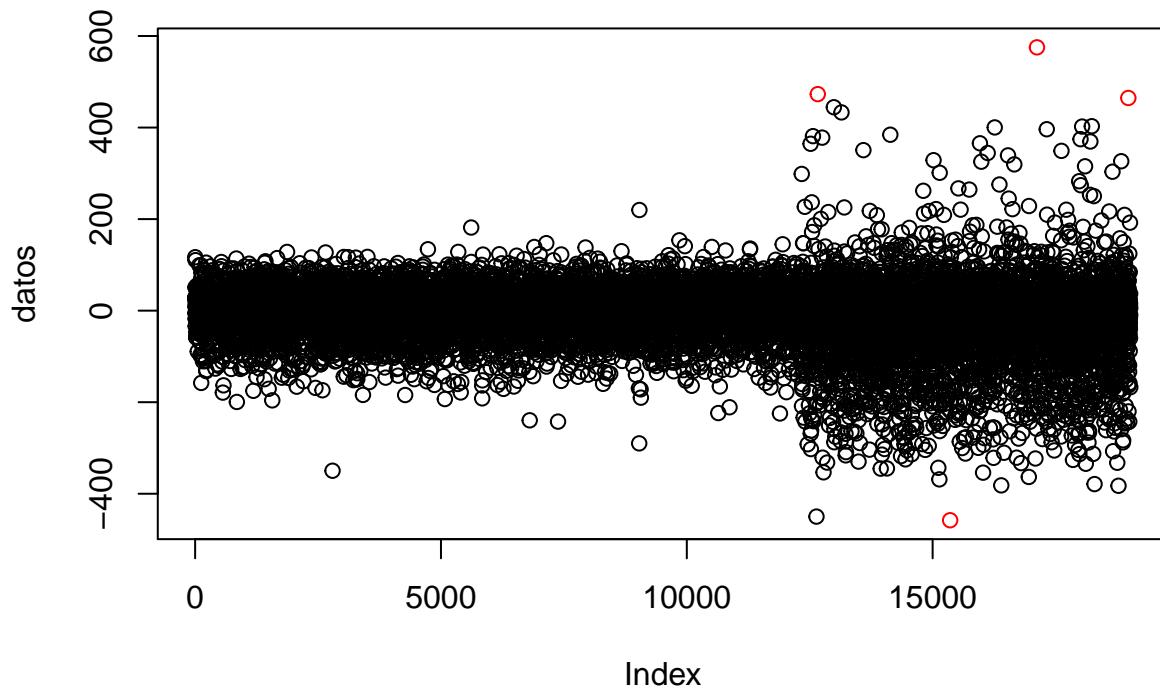
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 5267 5172 15482 13083  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

Test de Rosner



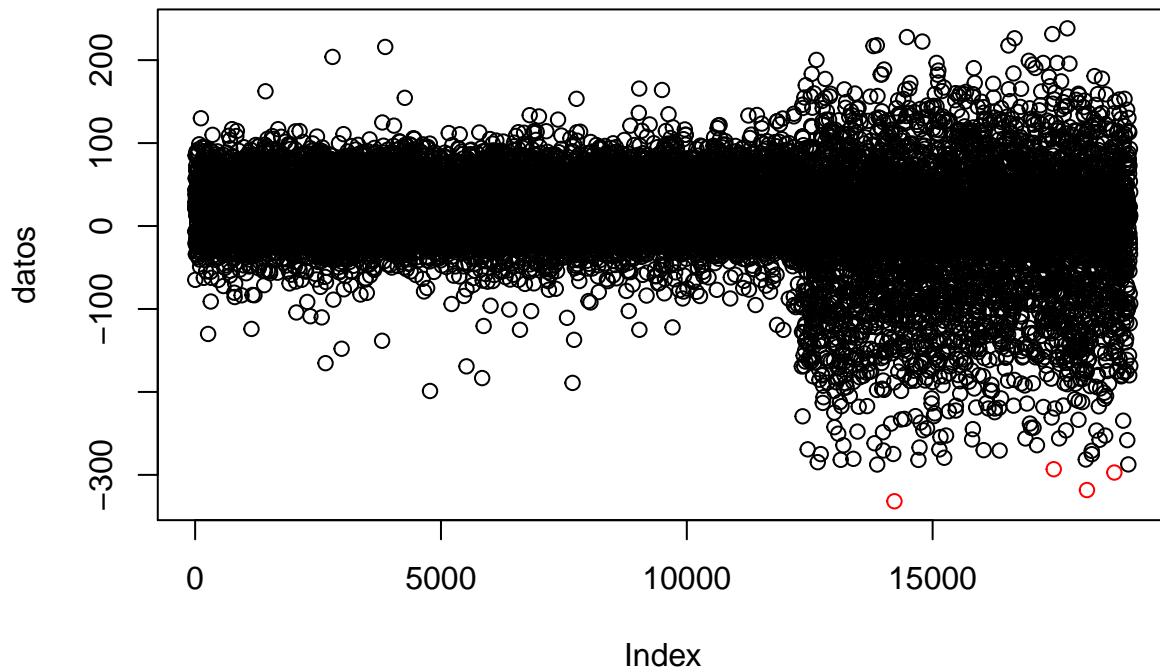
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17122 12664 18980 15356  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 17122 12664 18980 15356  
## Los valores de los outliers son: 575.2407 473.0654 464.631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

Test de Rosner



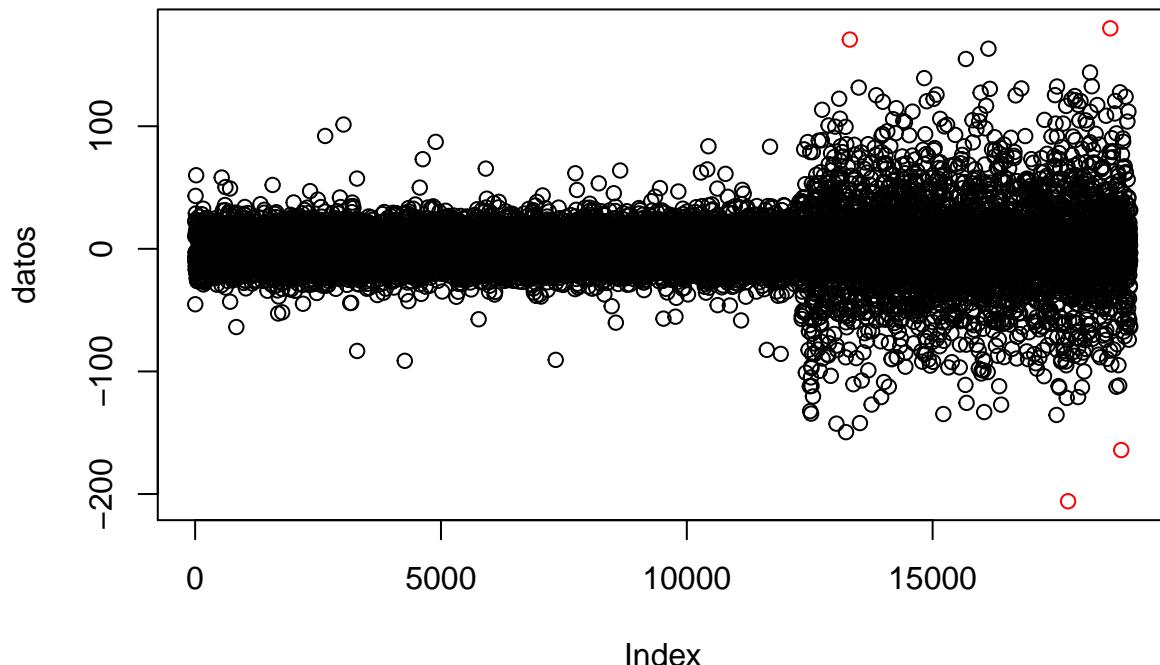
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 14223 18139 18696 17463  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 14223 18139 18696 17463  
## Los valores de los outliers son: -331.78 -318.3002 -297.1717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

Test de Rosner



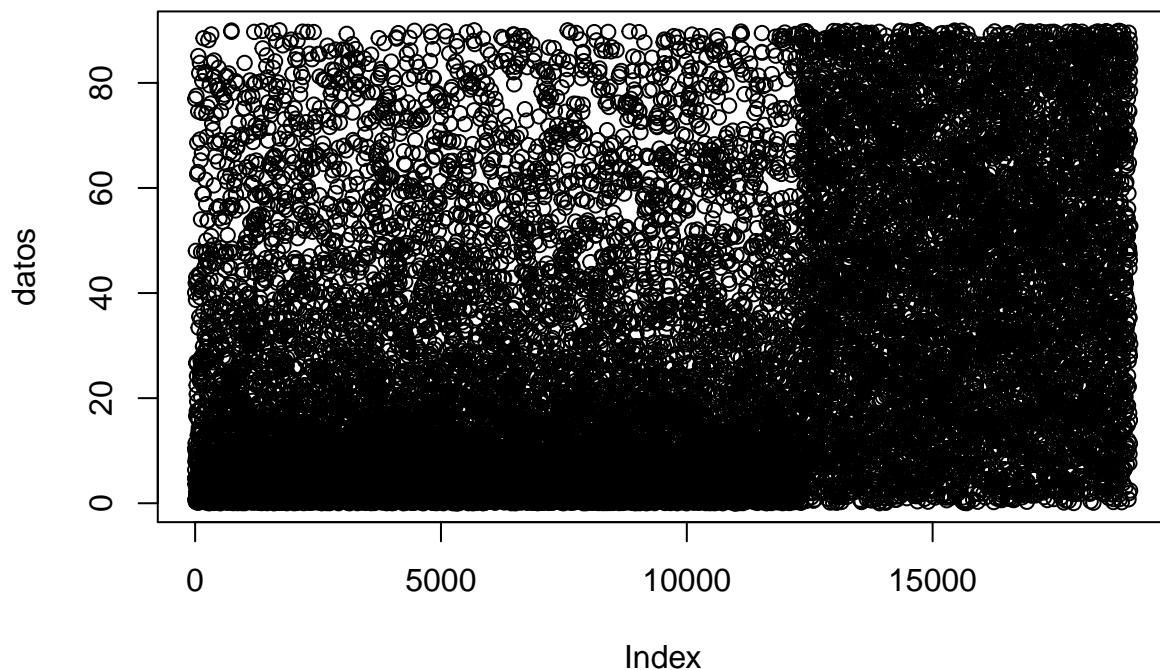
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 17755 18614 13316 18836  
## De las k mayores desviaciones, ?Qui?n es outlier? TRUE TRUE TRUE TRUE  
## Los ?ndices de los outliers son: 17755 18614 13316 18836  
## Los valores de los outliers son: -205.8947 179.851 170.692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

Test de Rosner



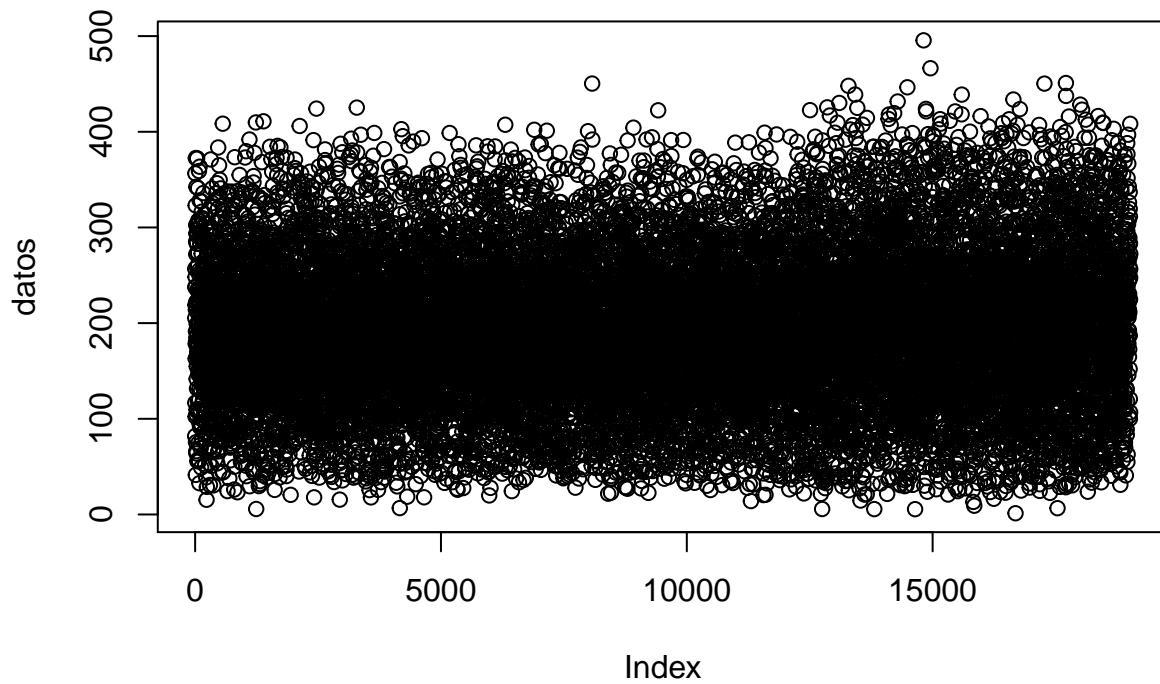
```
##  
## Test de Rosner  
## ?ndices de las k-mayores desviaciones de la media: 1714 5678 7577 14709  
## De las k mayores desviaciones, ?Qui?n es outlier? FALSE FALSE FALSE FALSE  
## Los ?ndices de los outliers son:  
## Los valores de los outliers son:
```

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.