

# 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.primerio = 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.primeros - (1.5*iqr)
extremo.inferior.outlier.extremo = cuartil.primeros - (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.superior.outlier.normal
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("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 7707
## 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)
```

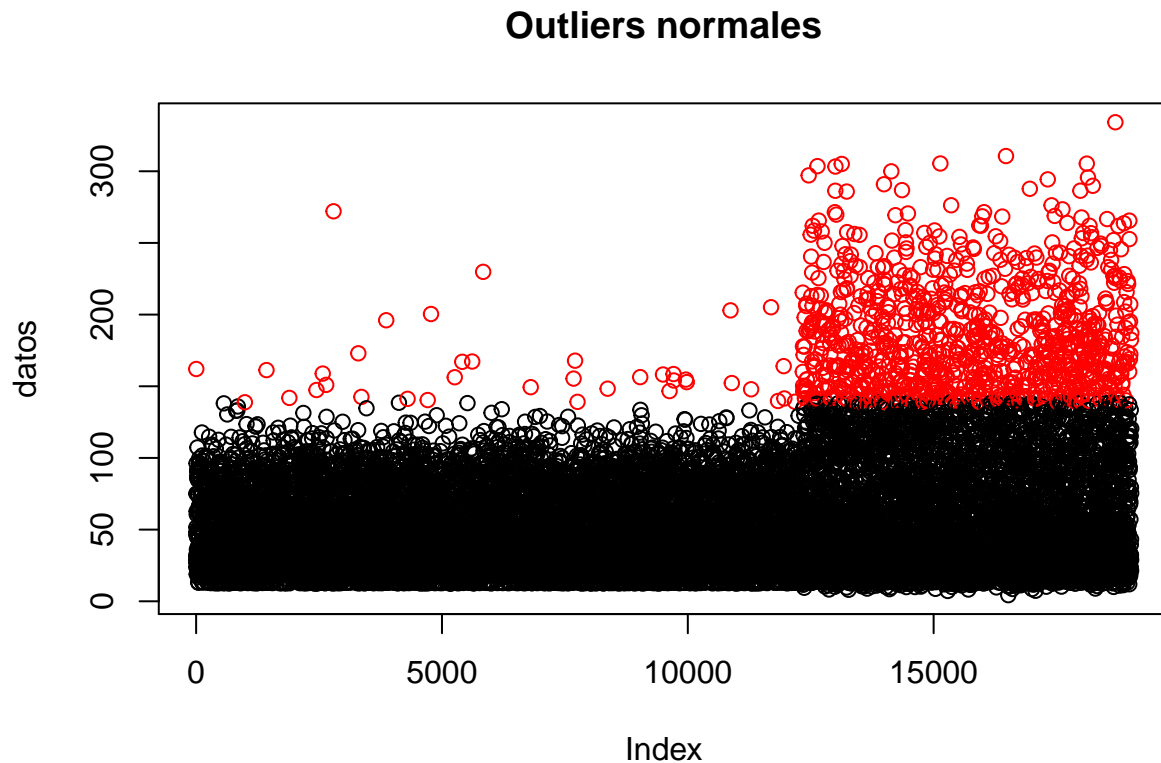
```
## [1] 0.8728221
```

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

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

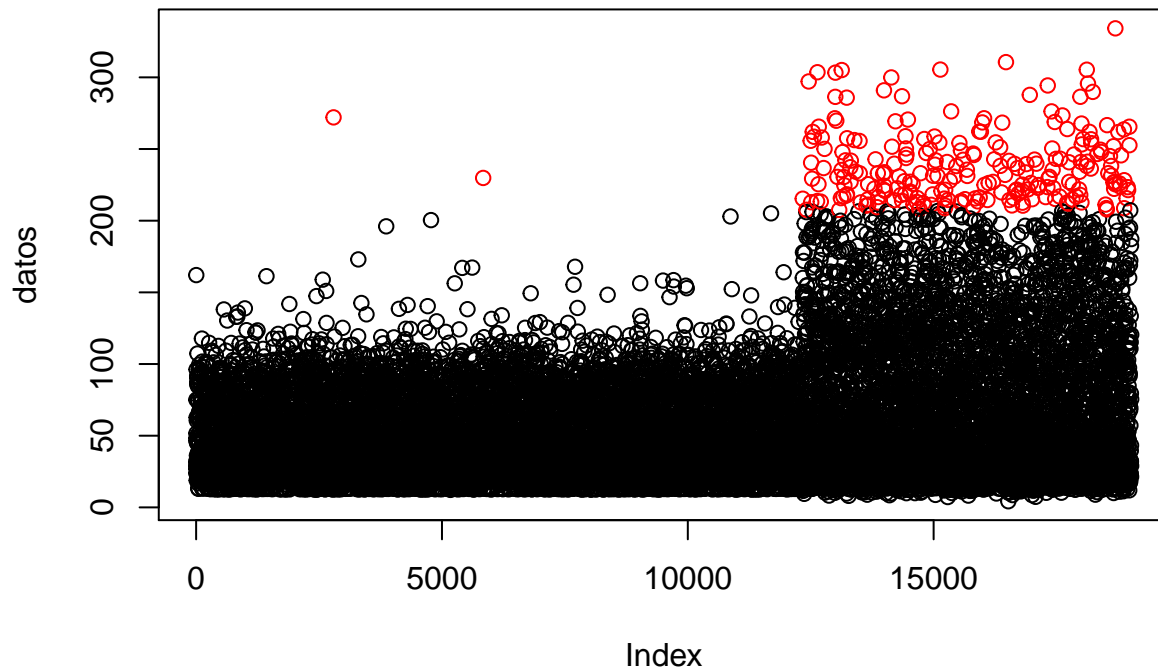
El siguiente paso será mostrar en un gráfico los datos que son outliers para identificarlos mejor, para ello haremos uso de la función *MiPlot\_Univariate\_Outliers()*.

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



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

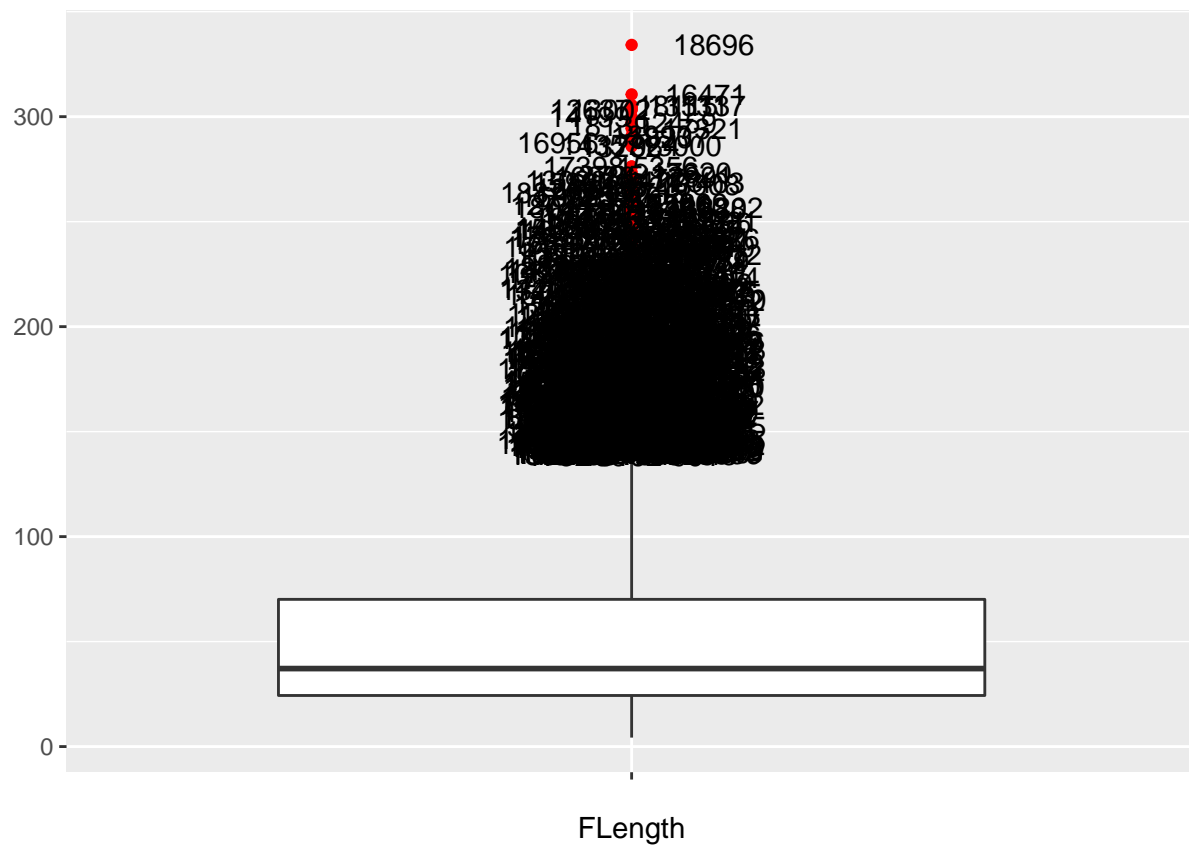
## Outliers extremos



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()`.

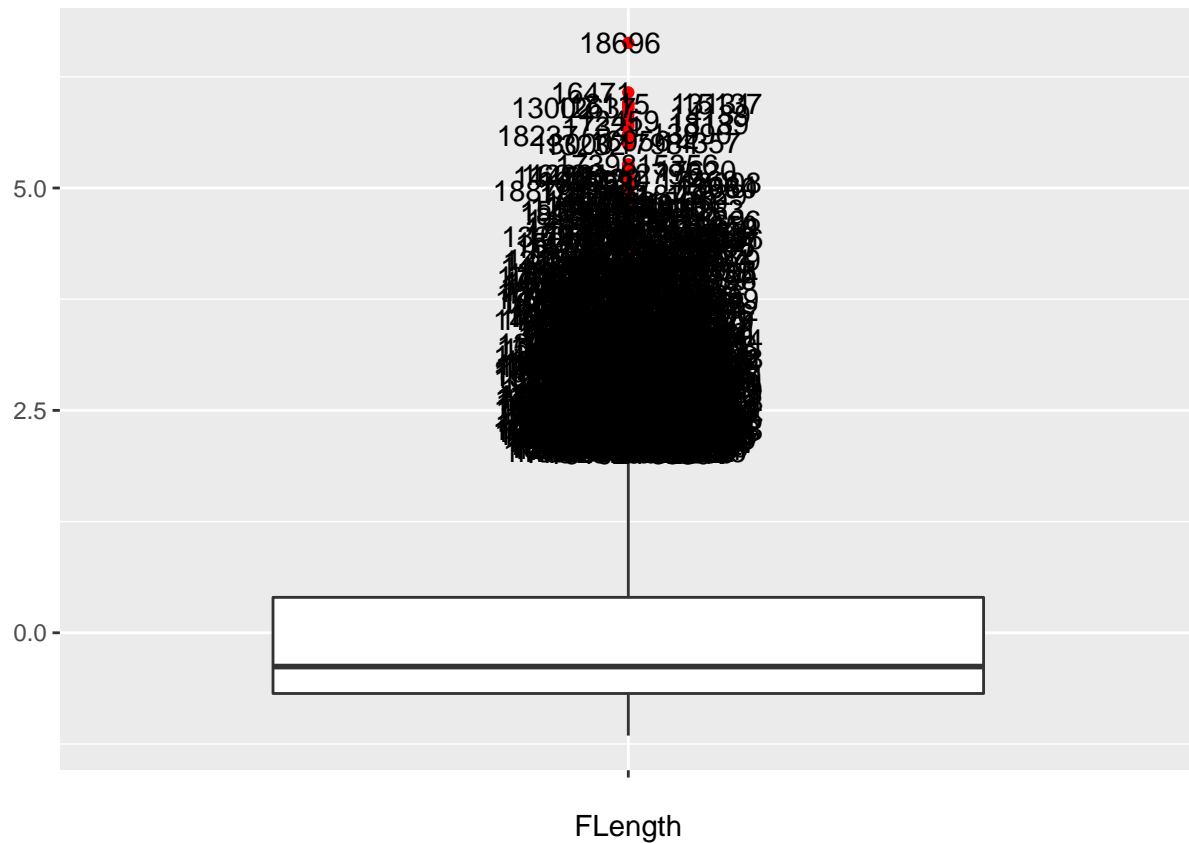
```
MiBoxPlot_IQR_Univariate_Outliers(mydata.numeric, indice.columna, coef=1.5)
```

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



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

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

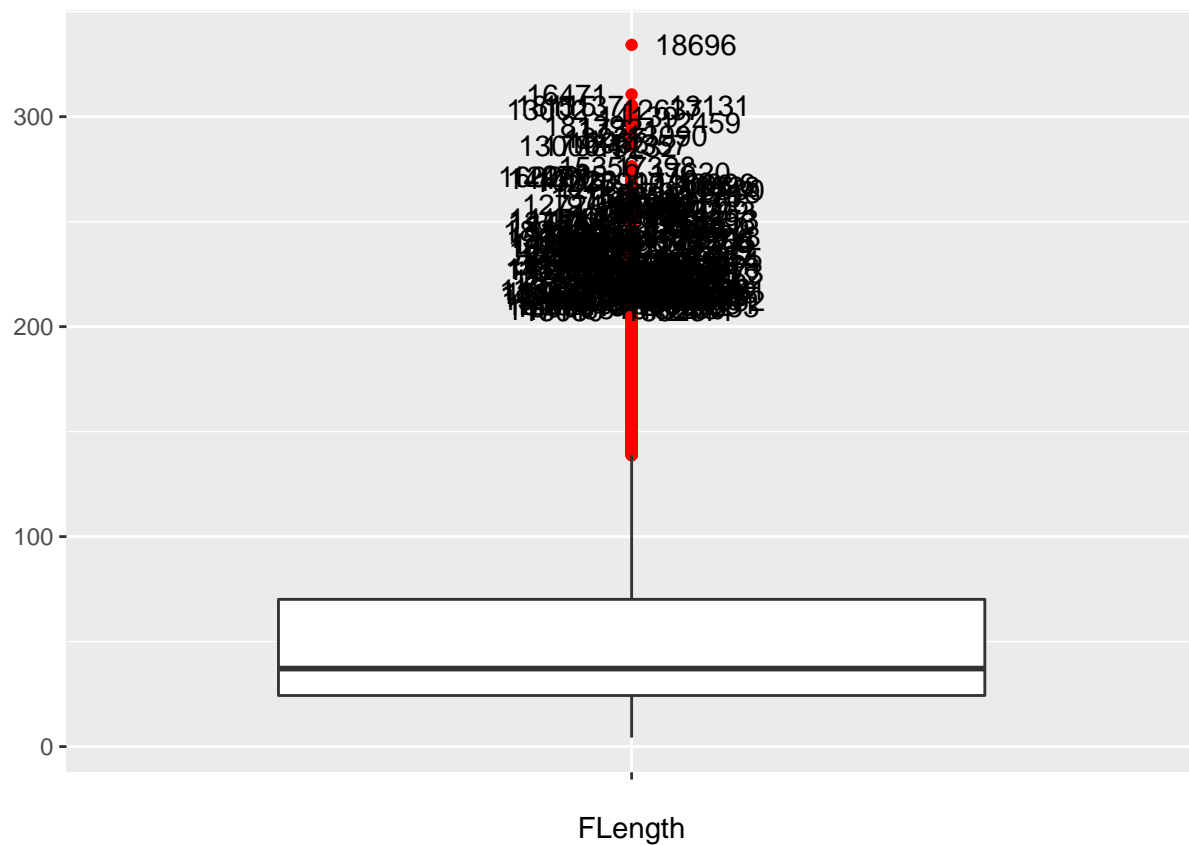


En este caso solamente lo utilizamos para los outliers normales, ya que la función de ggplot no nos permite expecificar un valor para calcular el outlier, utiliza directamente  $1.5 * IQR$ . Si quisiéramos ver donde están representados los datos , podríamos mostrar los labels de las instancias.

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

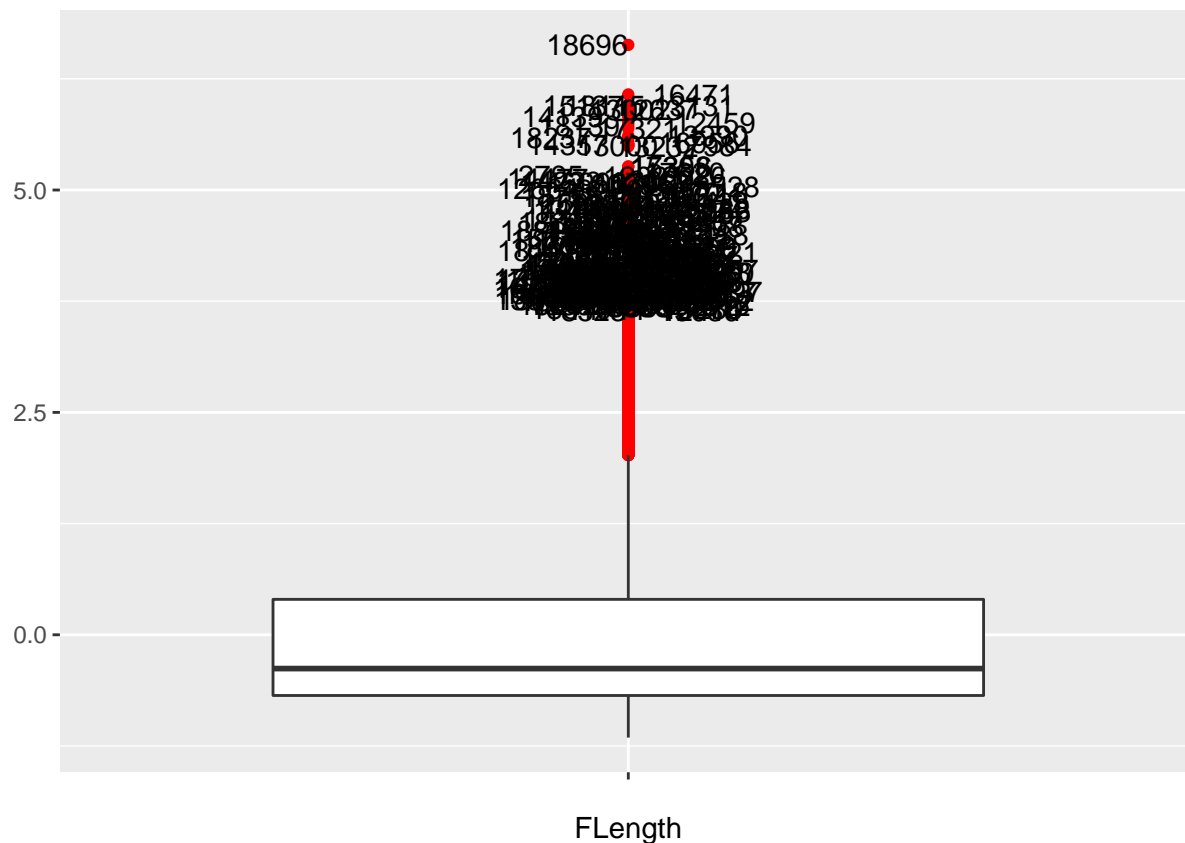
```
## [1] 236
```





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

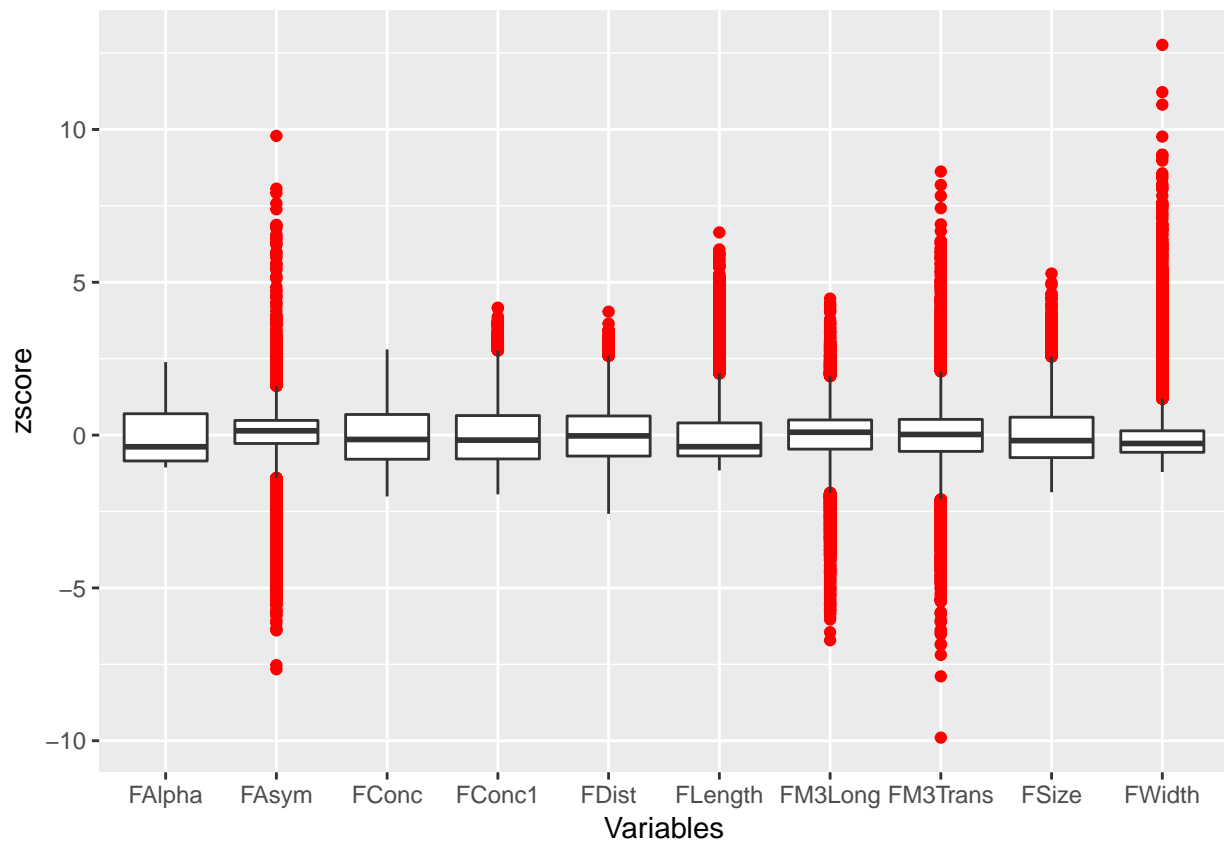
```
## [1] 236
```



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

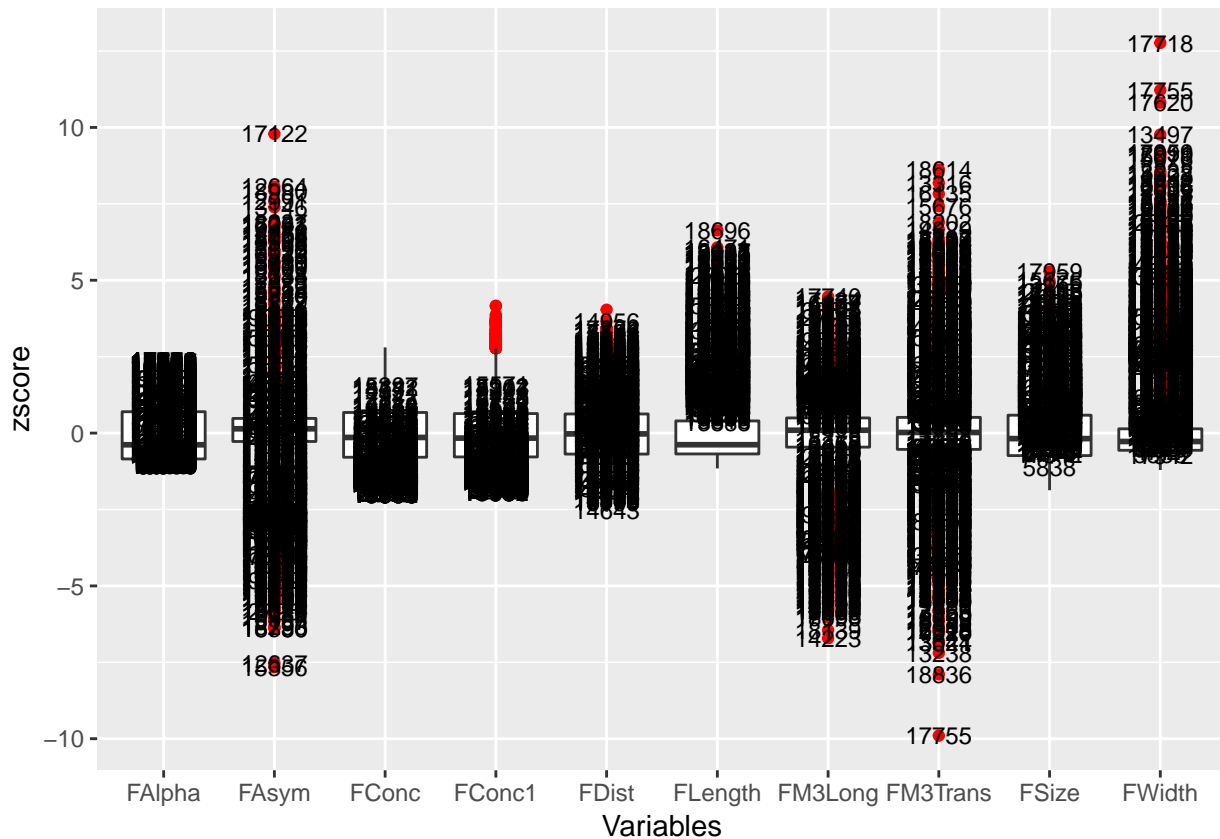
Lo siguiente que vamos a hacer es mirar que valores toman cada outlier en cada una de las columnas de nuestro dataset, así podemos apreciar si las instancias que toman valores anómalos en la columna que hemos estado estudiando toman también valores anómalos para otras columnas; para ello haremos uso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

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



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





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 ve 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
```