Trabajo Anomalías

Miguel Merelo Hernández

February 18, 2018

Crear datos

Utilizamos LetterRecognition del paquete mlbench. Solo utilizamos las primeras 200 líneas ya que la visualización de 20000 entradas hace imposible analizar el problema.

```
mydata.numeric.scaled<-scale(mydata.numeric,center=TRUE,scale=TRUE)
columna<-mydata.numeric[,indice.columna]
nombre.columna<-names(mydata.numeric[indice.columna])
columna.scaled<-scale(columna)
```

B1. UNIVARIATE STATISTICAL OUTLIERS -> IQR

1. Cómputo de los outliers IQR

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

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

```
## [1] 3
```

```
quantile(columna)
```

```
##   0%  25%  50%  75% 100%
##    1    3    4    6   12
```

Obtenemos que el 50% de los datos se encuentran entre los valores 3 y 6, que el valor máximo es 12 y el mínimo es 1. Con IQR=3 no podemos decir que la dispersión de los datos sea muy grande.

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

vector.es.outlier.normal <- columna>extremo.superior.outlier.normal |
  columna<extremo.inferior.outlier.normal
vector.es.outlier.extremo <- columna>extremo.superior.outlier.extremo |
  columna<extremo.inferior.outlier.extremo


print(paste("Extremo outlier normal.",
            "Inferior:",extremo.inferior.outlier.normal,
            "Superior:",extremo.superior.outlier.normal))
```

```
## [1] "Extremo outlier normal. Inferior: -1.5 Superior: 10.5"
```

```
print(paste("Extremo outlier extremo.",
            "Inferior:",extremo.inferior.outlier.extremo,
            "Superior:",extremo.superior.outlier.extremo))
```

```
## [1] "Extremo outlier extremo. Inferior: -6 Superior: 15"
```

```
print(paste("Indices de outliers normales:",
            paste(which(vector.es.outlier.normal),collapse = " ")))
```

```
## [1] "Indices de outliers normales: 10 28"
```

```
print(paste("Indices de outliers extremos:",
            paste(which(vector.es.outlier.extremo), collapse = " ")))
```

```
## [1] "Indices de outliers extremos: "
```

Consideramos como outlier normal a aquella entrada con valor menor a -1.5 o superior a 10.5 y como outlier extremo a aquellos con valores menores a -6 y superiores a 15. Para nuestros datos solo tenemos 2 valores de outliers normales y 0 extremos. Confirmamos que la dispersión de los datos es pequeña con solo un 1% de outliers.

Índices y valores de los outliers

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

```
## [1] 11 12
```

Obtenemos que los dos únicos outliers de la columna que habíamos seleccionado tienen los valores 11 y 12.

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

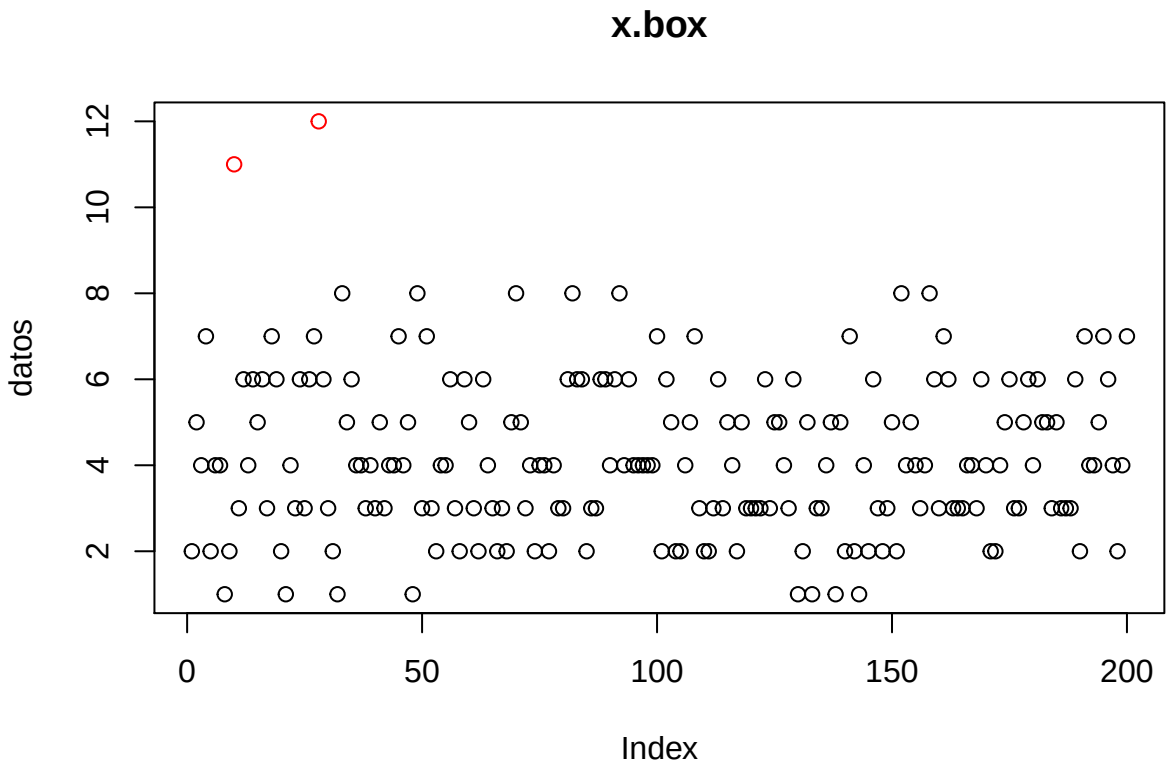
```
valores.normalizados.outliers.normales<-columna.scaled[vector.es.outlier.normal]
valores.normalizados.outliers.normales
```

```
## [1] 3.610215 4.141129
```

Valores de los outliers en el vector normalizado.

Plot

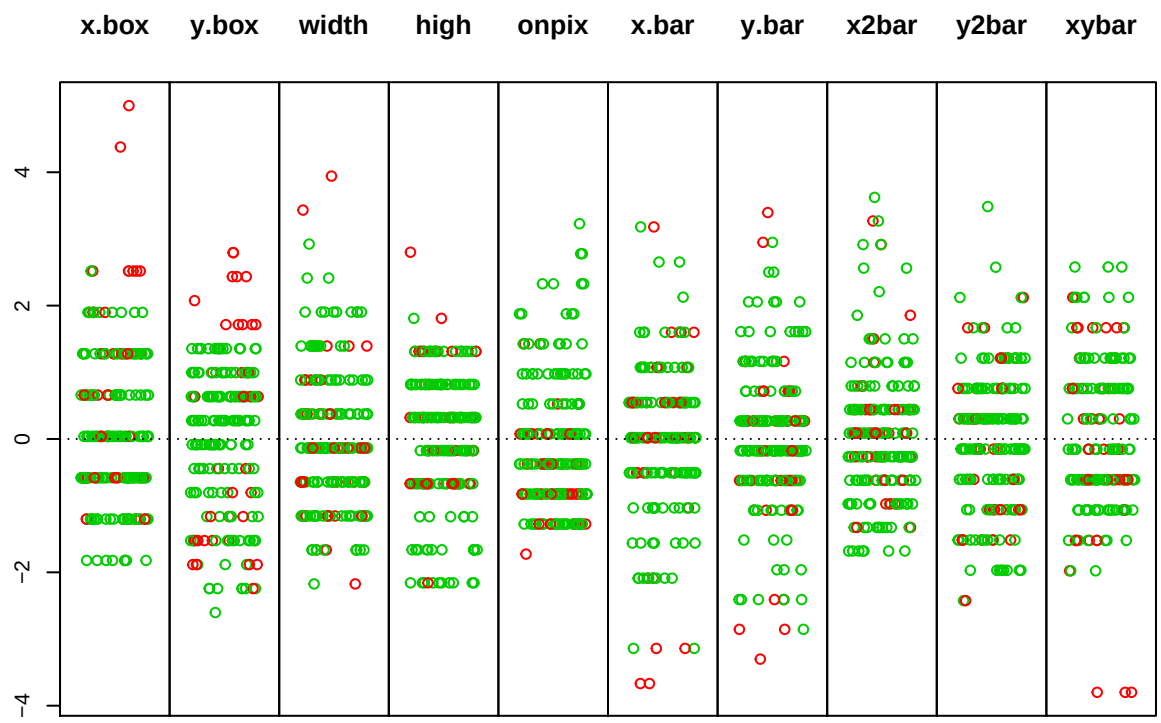
```
MiPlot_Univariate_Outliers(columna,claves.outliers.normales,nombre.columna)
```



C1. MULTIVARIATE STATISTICAL OUTLIERS -> Multivariate Normal Distribution -> Mahalanobis

1. Obtención de los outliers multivariantes

```
alpha.value = 0.05
alpha.value.penalizado = 1 - ( 1 - alpha.value) ^ (1 / nrow(mydata.numeric))
set.seed(12)
#solo con las 10 primeras variables, uni.plot no permite mas
mvoutlier.plot<-uni.plot(mydata.numeric[1:10],symb=FALSE,alpha=alpha.value.penalizado)
```



En el gráfico tenemos representados valores escalados de las variables con los que son outliers multivariantes.

2. Análisis de los outliers

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

```
## [1] 53
```

En nuestro conjunto de datos tenemos 53 outliers multivariantes.

```
indices.de.outliers.en.alguna.columna<-
  vector_claves_outliers_IQR_en_alguna_columna(mydata.numeric)
indices.de.outliers.en.alguna.columna<-
  indices.de.outliers.en.alguna.columna[!duplicated(indices.de.outliers.en.alguna.columna)]
indices.de.outliers.multivariantes.MCD<-which(is.MCD.outlier)
indices.de.outliers.multivariantes.MCD.pero.no.1variantes<-
  setdiff(indices.de.outliers.multivariantes.MCD,indices.de.outliers.en.alguna.columna)
nombres.de.outliers.multivariantes.MCD.pero.no.1variantes<-
  names(is.MCD.outlier[indices.de.outliers.multivariantes.MCD.pero.no.1variantes])
indices.de.outliers.multivariantes.MCD.pero.no.1variantes
```

```
##  [1]   4   9  12  16  18  45  49  56  59  63  70  71  82 108 123 127 132
## [18] 152 156 179 184 189 190 200
```

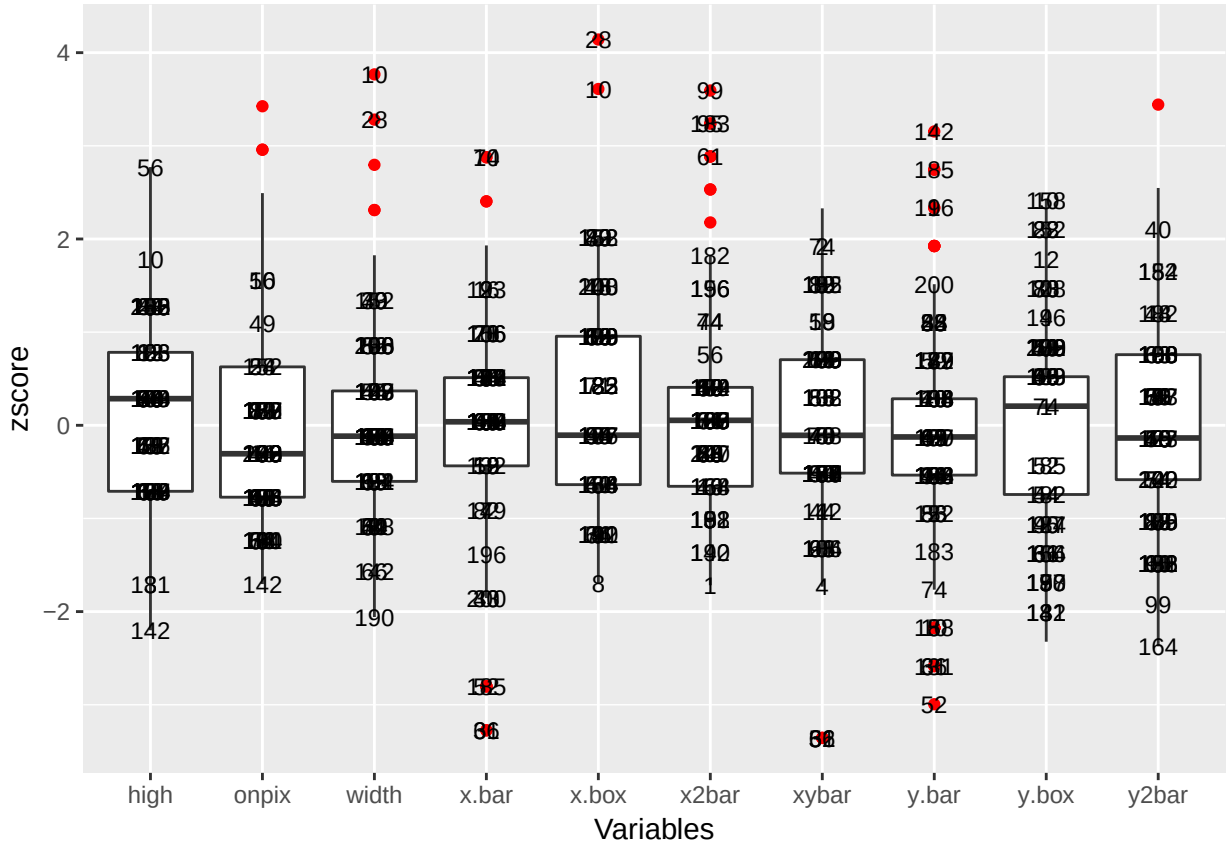
Indices de los outliers que son únicamente multivariantes.

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

```
##        x.box      y.box      width       high      onpix      x.bar
## 1 -1.1680108  0.2054706 -1.0873763 -0.2113252 -1.2380388  0.5110552
## 2  0.4247312  1.4699052 -1.0873763  0.7831463 -0.7717341  1.4574537
## 4  1.4865592  1.1537966  0.3689313  0.2859105 -0.3054295 -0.9085426
## 7 -0.1061828 -1.6911813 -0.1165046 -0.7085609  0.1608751  0.5110552
## 8 -1.6989248 -2.0072899 -1.0873763 -1.7030323 -1.2380388  0.5110552
## 9 -1.1680108 -1.6911813 -0.6019405 -0.7085609 -0.7717341  1.4574537
##        y.bar       x2bar      y2bar      xybar      x2ybr        xy2br
## 1  2.3343823 -1.71716256  0.3109080 -0.9197064  1.3318240 -0.002417871
## 2 -0.9448202  0.05310812 -0.5837913  1.9226533 -1.2578338  0.481156231
## 4  0.6947810 -0.30094601  0.3109080 -1.7318092 -0.8878827  0.964730333
## 7 -0.1250196  0.40716226  0.3109080 -0.5136550 -0.1479804 -0.969566075
## 8 -2.1745212 -1.00905429 -1.4784905 -0.1076036 -1.6277849 -0.002417871
## 9 -0.5349199 -1.00905429  0.3109080  1.5166019 -0.8878827 -0.002417871
##        x.ege       xegvy       y.ege      yegvx
## 1 -1.3059334 -0.09718502 -1.45895955  0.1809413
## 2 -0.4537746 -0.09718502  0.07678734  1.4288120
## 4  1.2505431  1.15681519 -0.69108610  0.1809413
## 7 -0.4537746 -0.09718502  1.22859751  1.4288120
## 8 -0.8798540 -1.35118522 -0.69108610 -0.4429941
## 9 -0.8798540 -1.35118522 -1.07502282 -0.4429941
```

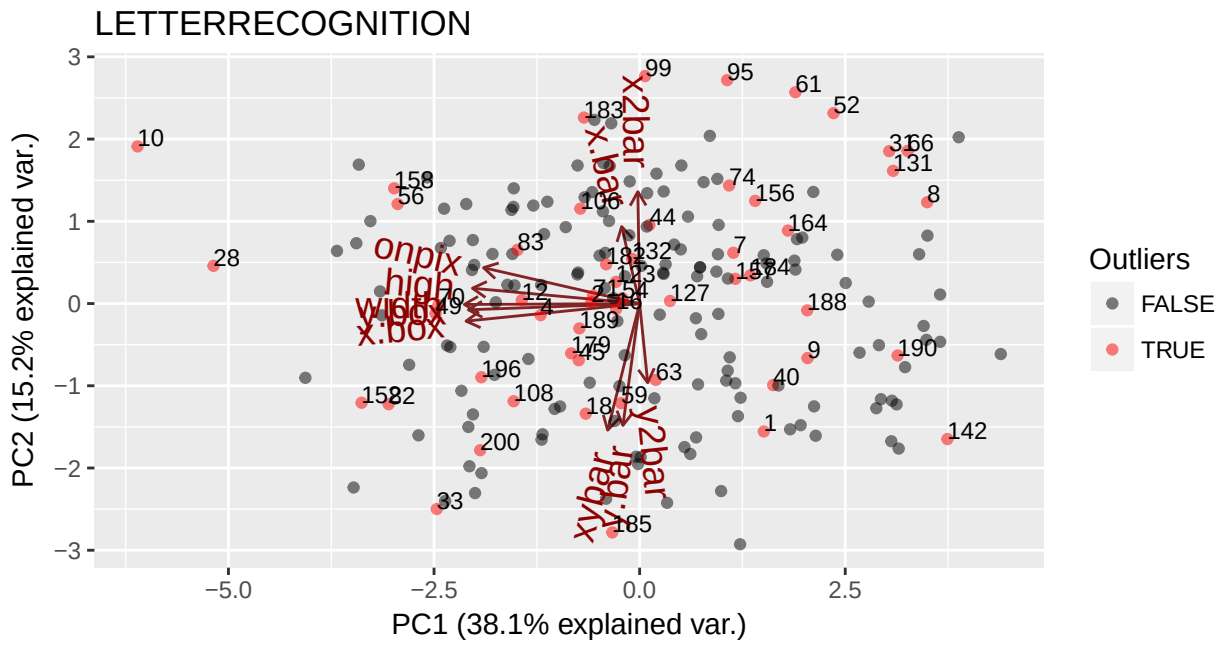
Valores normalizados de las instancias con outliers.

```
set.seed(12)
MiBoxPlot_juntos(mydata.numeric[1:10],is.MCD.outlier)
```



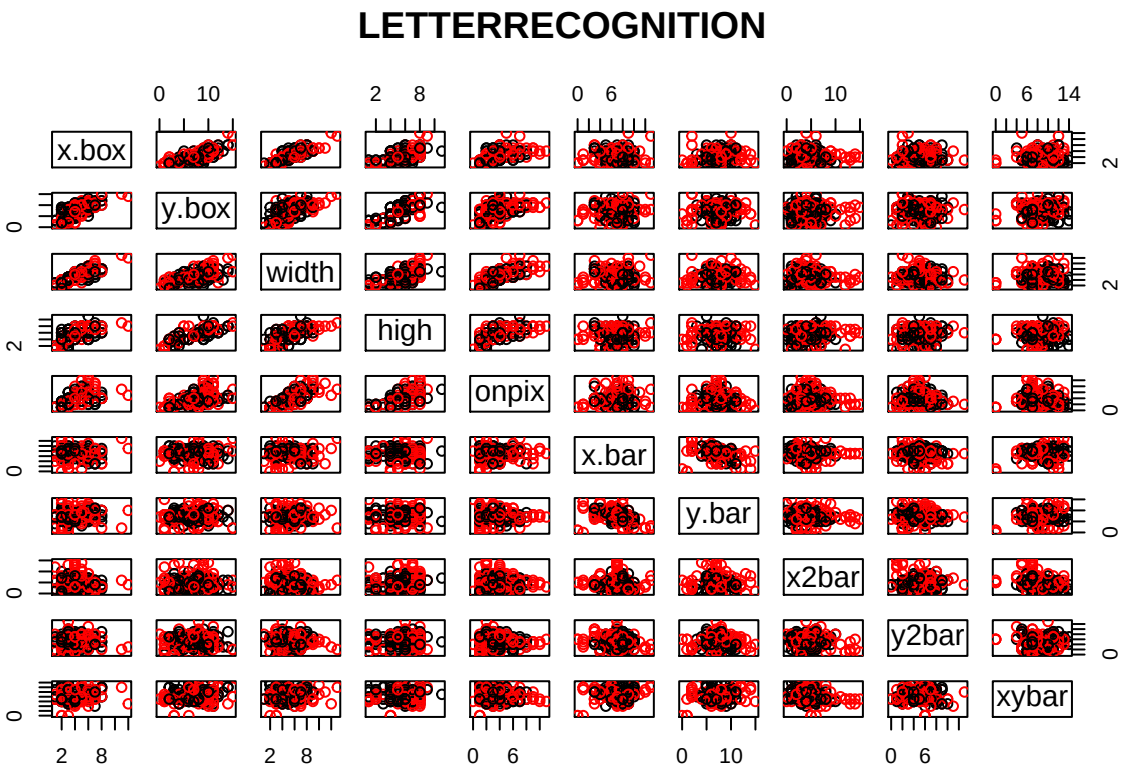
Podemos ver como se muestran en rojo las instancias con outliers simples y multivariables.

```
set.seed(12)
MiBiPlot_Multivariate_Outliers(mydata.numeric[1:10],is.MCD.outlier,"LETTERRECOGNITION")
```



En este gráfico se colorean en rojo aquellos valores que son outliers multivariantes.

```
set.seed(12)
MiPlot_Univariate_Outliers(mydata.numeric[1:10],
                           indices.de.outliers.en.alguna.columna,
                           "LETTERRECOGNITION")
```

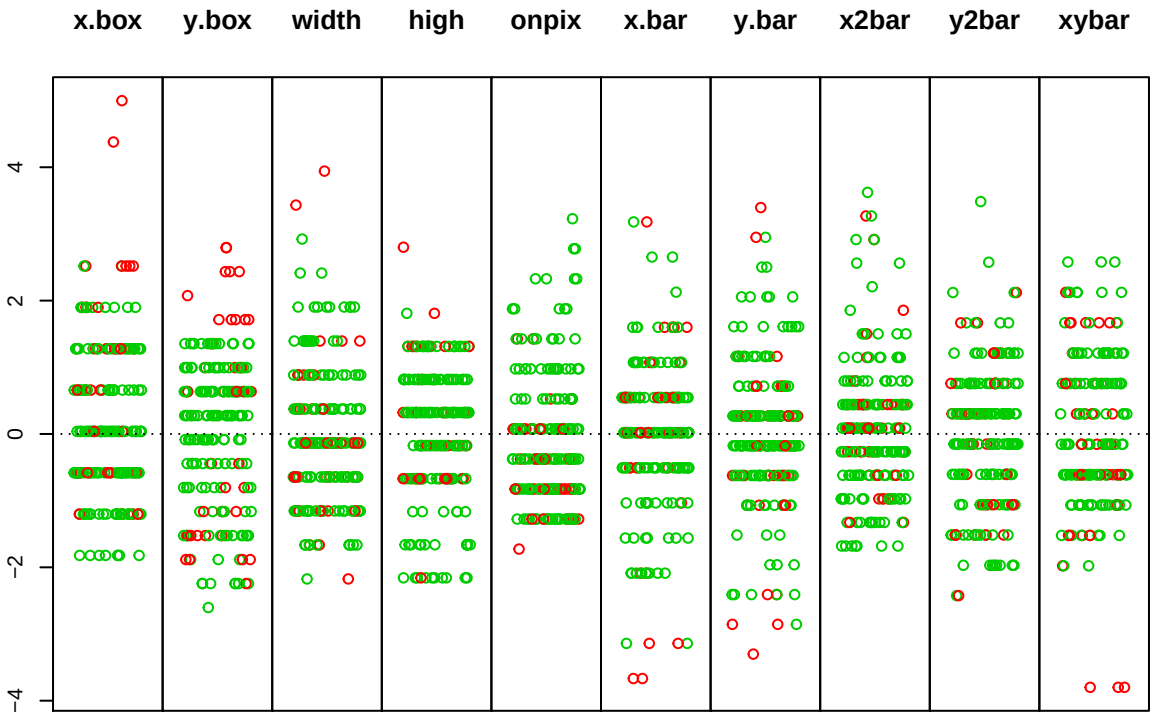


Mostramos las variables dos a dos y marcamos aquellos puntos pertenecientes a instancias que tengan algún outlier en alguna variable. Por la cantidad de instancias con outliers que tenemos, casi el 50%, es imposible sacar una conclusión de este gráfico.

D1. MULTIVARIATE STATISTICAL OUTLIERS -> LOF

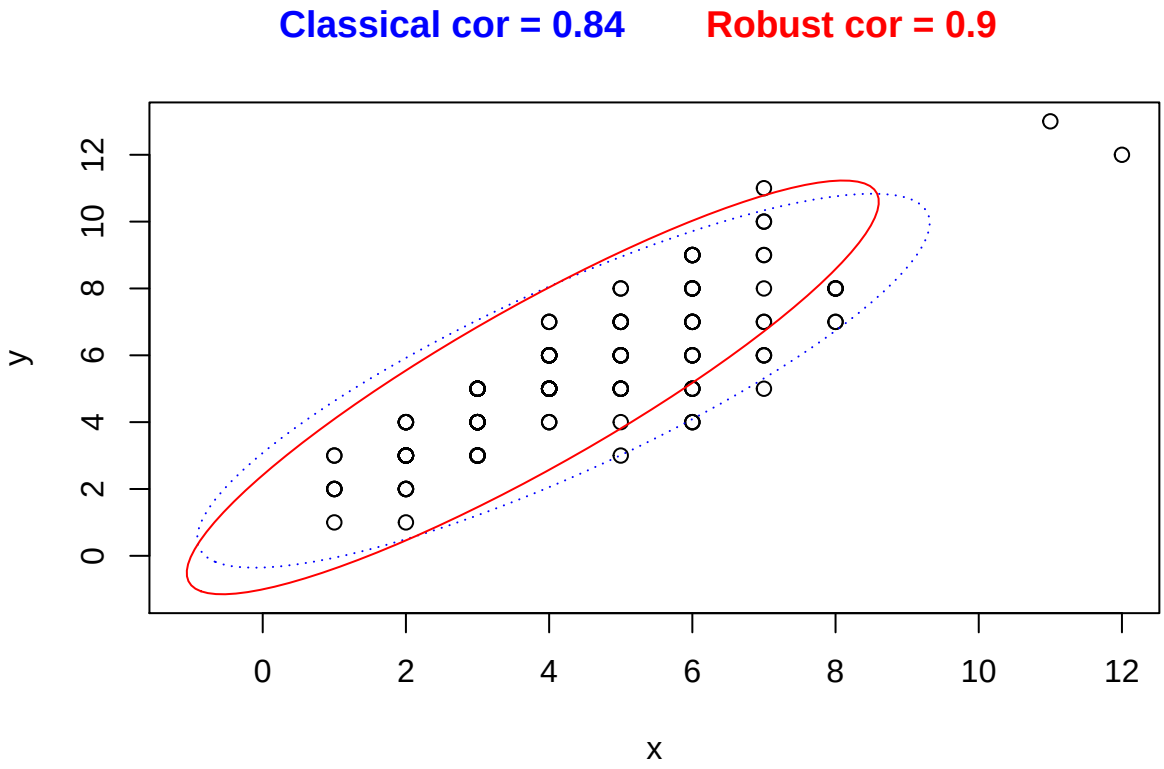
```
mis.datos.numericos<-LetterRecognition[1:200,-c(1)]
mis.datos.numericos.normalizados<-scale(mis.datos.numericos)
row.names(mis.datos.numericos.normalizados)<-row.names(mis.datos.numericos)
```

```
alpha.value = 0.05
alpha.value.penalizado = 1 - ( 1 - alpha.value) ^ (1 / nrow(mis.datos.numericos))
set.seed(12)
mvoutlier.plot<-uni.plot(mis.datos.numericos[1:10],
                         symb=FALSE,
                         alpha=alpha.value.penalizado)
```



Tenemos el mismo gráfico ya analizado con los outliers multivariables marcados en rojo.

```
is.MCD.outlier<-mvoutlier.plot$outliers
numero.de.outliers.MCD<-sum(is.MCD.outlier)
corr.plot(mis.datos.numericos[,1], mis.datos.numericos[,3])
```

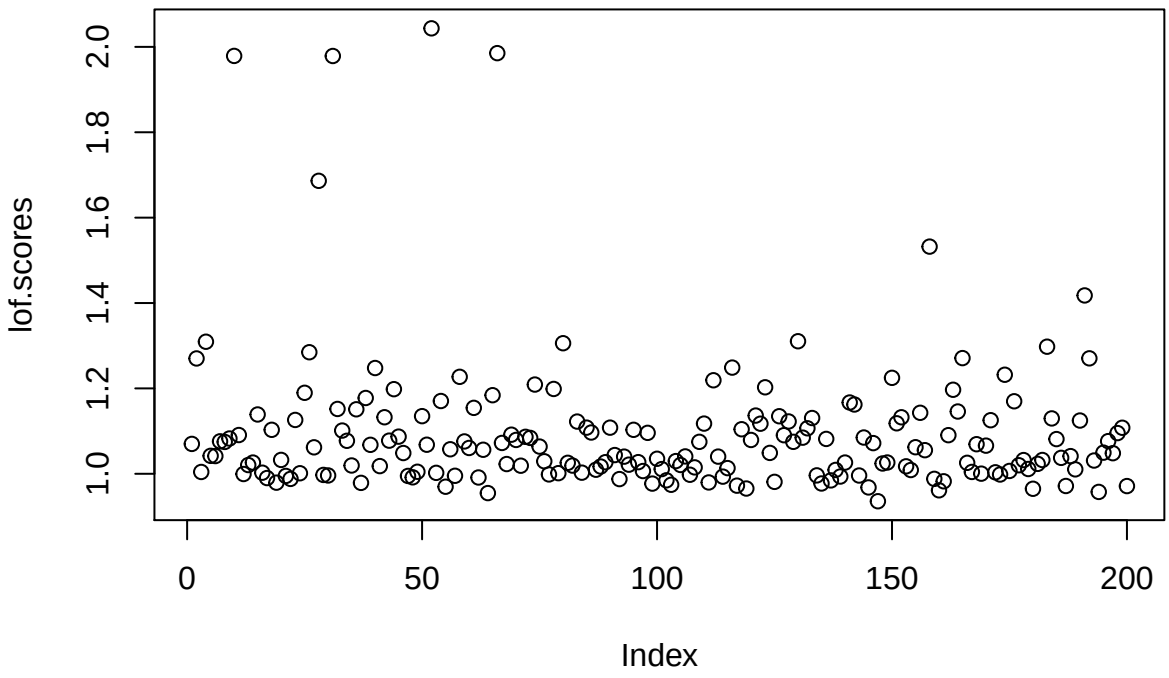



```
## $cor.cla
## [1] 0.8436275
##
## $cor.rob
## [1] 0.8972285
```

Entre estas dos variables podemos ver como hay dos outliers bastante claros con valores 11 y 12 en X y 13 y 12 en Y.

1. DISTANCE BASED OUTLIERS (LOF)

```
numero.de.vecinos.lof = 5
set.seed(12)
lof.scores<-lofactor(mis.datos.numericos.normalizados,numero.de.vecinos.lof)
plot(lof.scores)
```

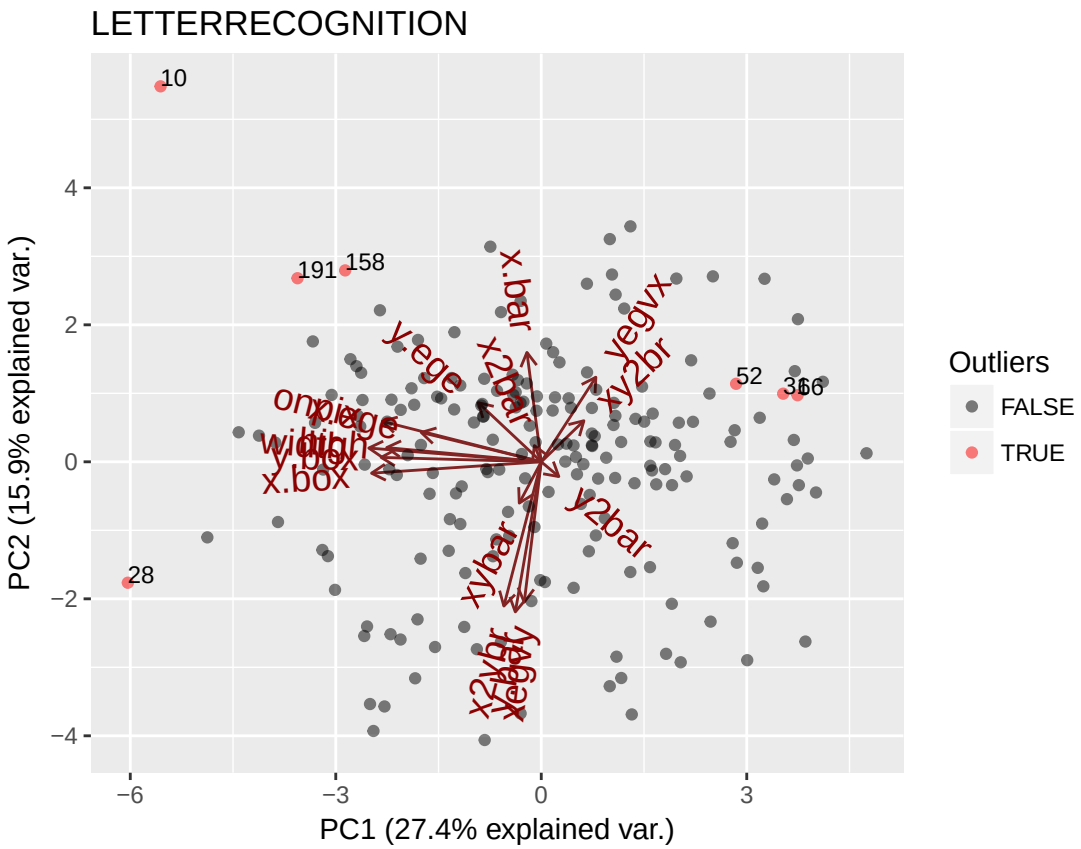


Vemos que hay 5 puntos con un lof claramente más alto que los demás, por encima de 1.6 y otros 2 que destacan entre 1.4 y 1.6 por lo que fijaremos el número de outliers en 7.

```
numero.de.outliers = 7
indices.de.lof.outliers.ordenados<-order(lof.scores,decreasing=TRUE)
indices.de.lof.top.outliers<-indices.de.lof.outliers.ordenados[1:numero.de.outliers]
is.lof.outlier<-row.names(mis.datos.numericos) %in% indices.de.lof.top.outliers
indices.de.lof.top.outliers
```

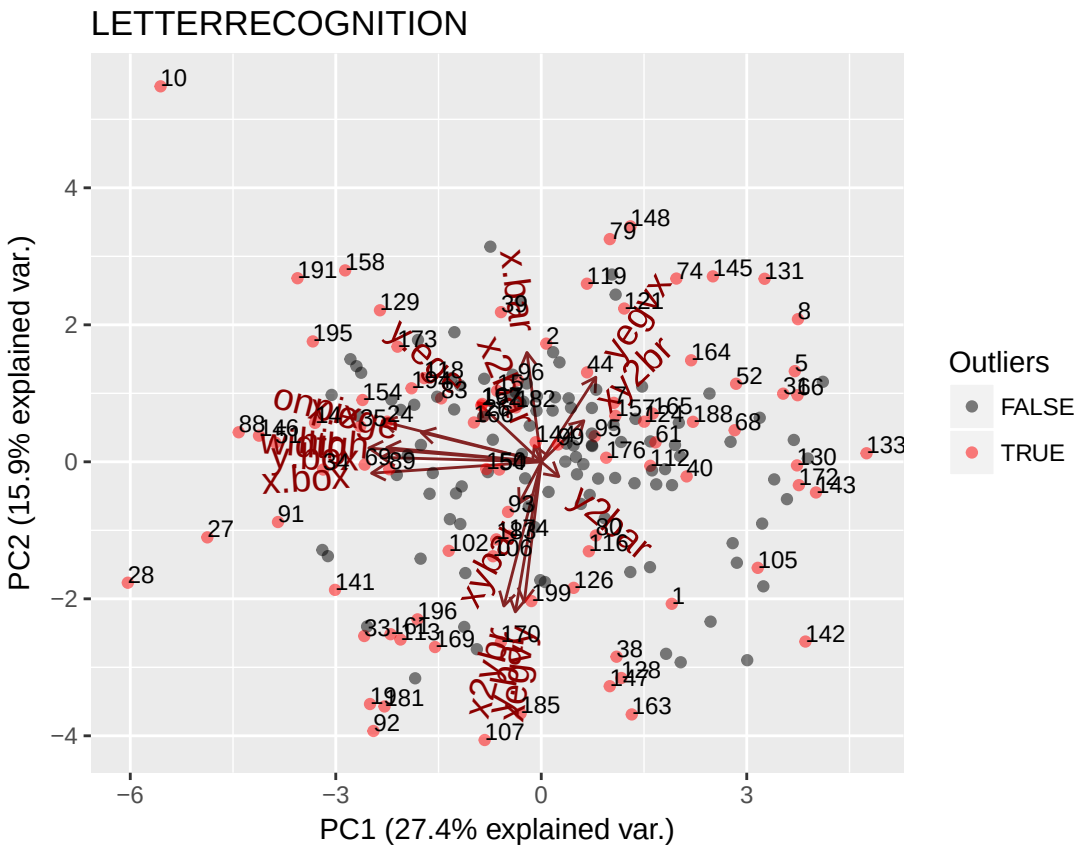
```
## [1]  52  66  10  31  28 158 191
```

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



Los 5 puntos con mayor lof son 10, 28, 52, 31 y 66 mientras que los dos que consideramos outliers por superar el valor 1.4 de lof son 191 y 158.

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



Vemos que los puntos con mayor lof que habíamos seleccionado son outliers por columna.

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

```
##  [1]   1   2   5   7   8  14  15  19  24  26  27  33  34  35  38  39  40
## [18]  44  51  54  61  68  69  74  79  80  83  88  89  91  92  93  95  96
## [35]  99 102 105 106 107 112 113 116 118 119 121 124 126 128 129 130 131
## [52] 133 141 142 143 144 145 146 147 148 150 154 157 161 163 164 165 166
## [69] 167 169 170 172 173 174 176 181 182 183 185 188 192 194 195 196 199
```

Confirmamos la conclusión del gráfico anterior por lo que podemos decir que, para nuestro caso, usando distancia de Mahalanobis o usando LOF llegamos al mismo resultado.

```
data.frame.numeric<-LetterRecognition[sapply(LetterRecognition,is.numeric)]
head(data.frame.numeric)
```

```
##   x.box y.box width high onpix x.bar y.bar x2bar y2bar xybar x2ybr xy2br
## 1     2     8     3    5     1     8    13     0     6     6    10     8
## 2     5    12     3    7     2    10     5     5     4    13     3     9
## 3     4    11     6    8     6    10     6     2     6    10     3     7
## 4     7    11     6    6     3     5     9     4     6     4     4    10
## 5     2     1     3    1     1     8     6     6     6     6     5     9
## 6     4    11     5    8     3     8     8     6     9     5     6     6
##   x.ege xegvy y.ege yegvx
## 1     0     8     0     8
## 2     2     8     4    10
## 3     3     7     3     9
## 4     6    10     2     8
```

```
## 5     1     7     5    10
## 6     0     8     9     7
```

D2. MULTIVARIATE STATISTICAL OUTLIERS. CLUSTERING OUTLIERS

Para mejorar la visualización de los datos, solo vamos a considerar las instancias con las 3 primeras letras, A, B y C, de nuestros datos. Por ello fijaremos el número del cluster en 3

```
library(mlbench)
data("LetterRecognition")
mis.datos.numericos<-LetterRecognition[which(LetterRecognition$lettr %in% LETTERS[1:3]),-c(1)]
mis.datos.numericos<-mis.datos.numericos[1:200,]
mis.datos.numericos.normalizados<-scale(mis.datos.numericos)
row.names(mis.datos.numericos.normalizados)<-row.names(mis.datos.numericos)
numero.de.outliers   = 7
numero.de.clusters   = 3

set.seed(2)
```

```
result<-kmeans(mis.datos.numericos.normalizados,numero.de.clusters)
indices.clustering.LetterRecognition<-result$cluster
centroides.normalizados.LetterRecognition<-result$centers
head(indices.clustering.LetterRecognition)
```

```
##  7  8 18 26 34 59
##  1  3  2  2  2  2
```

Creamos los clusters y se asocia cada instancia a un cluster.

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

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

```
dist.centroides.LetterRecognition<-
  distancias_a_centroides(mis.datos.numericos.normalizados,
                          indices.clustering.LetterRecognition,
                          centroides.normalizados.LetterRecognition)
top.outliers.LetterRecognition<-
  order(dist.centroides.LetterRecognition,decreasing=TRUE)[1:numero.de.outliers]
top.outliers.LetterRecognition
```

```
## [1]  52  21 127 119  96   4 141
```

Las instancias con mayor distancia al centroide son 52, 21, 127, 119, 96, 4 y 141 y serán las que quizás podríamos cambiar de cluster ya que el objetivo que tenemos es el de minimizar la distancia de cada instancia con su centroide.

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

```
  dist.centroides<-distancias_a_centroides(datos.normalizados,
                                           indices.asignacion.clustering,
                                           datos.centroides.normalizados)
  respuesta<-list()
  dist.centroides.sorted<-order(dist.centroides, decreasing = TRUE)
  respuesta$indices<-head(dist.centroides.sorted,n=numero.de.outliers)
  respuesta$distancias<-dist.centroides[respuesta$indices]
  respuesta
}
top.outliers.kmeans<-top_clustering_outliers(mis.datos.numericos.normalizados,
                                             indices.clustering.LetterRecognition,
                                             centroides.normalizados.LetterRecognition,
                                             numero.de.outliers)
top.outliers.kmeans$indices
```

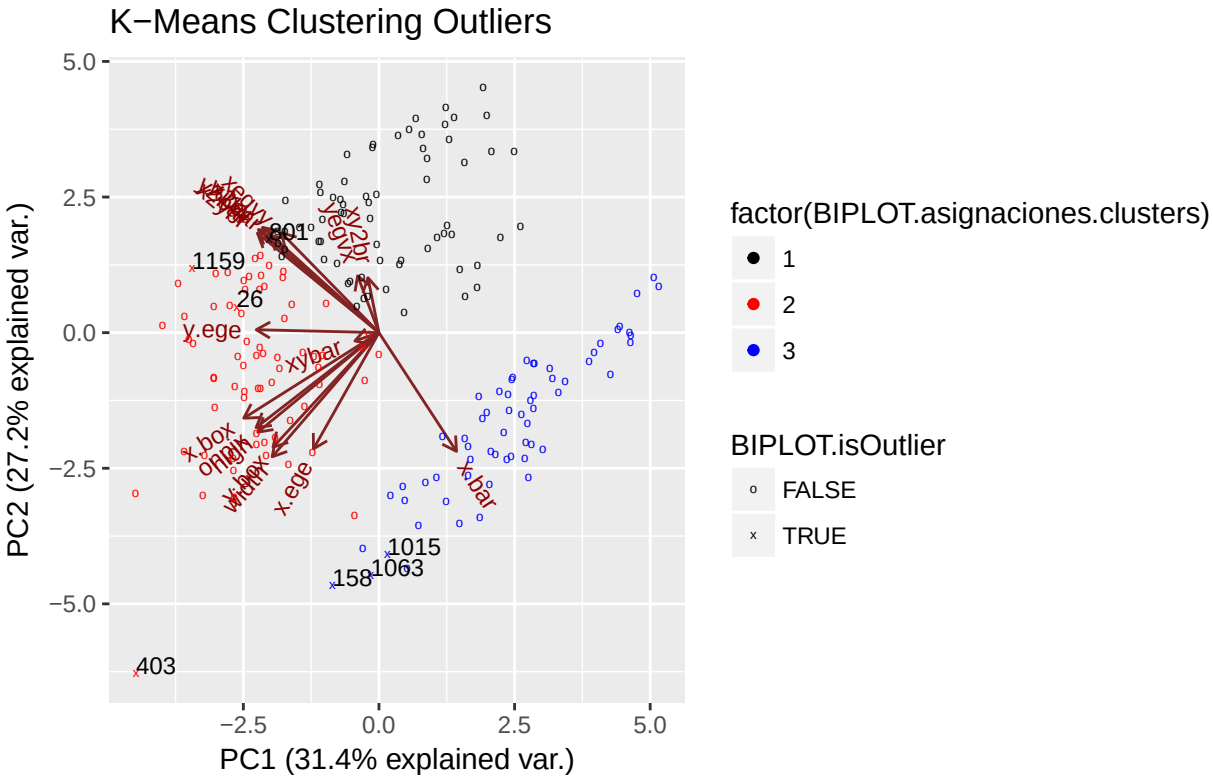
```
## [1]  52  21 127 119  96   4 141
```

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

```
##      403      158     1063     1015      801       26     1159
## 7.311972 5.704498 5.536308 5.211792 4.756320 4.714793 4.690884
```

Obtenemos el mismo resultado que previamente junto a la distancia.

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


BIPLOT.isOutlier             = is.kmeans.outlier
BIPLOT.cluster.colors        = c("black","red","blue")
BIPLOT.asignaciones.clusters = indices.clustering.LetterRecognition
MiBiPlot_Clustering_Outliers(mis.datos.numericos, "K-Means Clustering Outliers")
```



Vemos como se colorean los puntos según al cluster al que pertenezcan y que hay algún punto que está muy cerca de otro cluster, como el que del cluster 2 que vemos cerca del cluster 3.

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

```
##       x.box    y.box    width     high    onpix    x.bar    y.bar    x2bar
## 1 2.705882 3.941176 3.470588 3.455882 2.029412 6.294118 7.397059 5.720588
## 2 5.071429 9.057143 6.042857 6.742857 5.171429 7.042857 7.214286 5.700000
## 3 3.032258 6.629032 4.741935 4.870968 2.306452 9.290323 3.096774 2.645161
##       y2bar    xybar    x2ybr    xy2br    x.ege    xegvy    y.ege    yegvx
## 1 6.705882 8.176471 6.573529 9.514706 1.455882 8.441176 4.500000 8.985294
## 2 5.900000 8.385714 6.371429 8.228571 3.485714 8.442857 6.242857 8.500000
## 3 1.790323 8.161290 1.774194 8.693548 2.354839 6.080645 2.435484 7.758065
```

Los valores de los centroides estaban normalizados. Revertimos la operación.

```
library(cluster)
mis.datos.numericos.dist<-dist(mis.datos.numericos.normalizados)
modelo.pam<-pam(mis.datos.numericos.dist,k=numero.de.clusters)
medoides.valores <- mis.datos.numericos[modelo.pam$medoids, ]
```

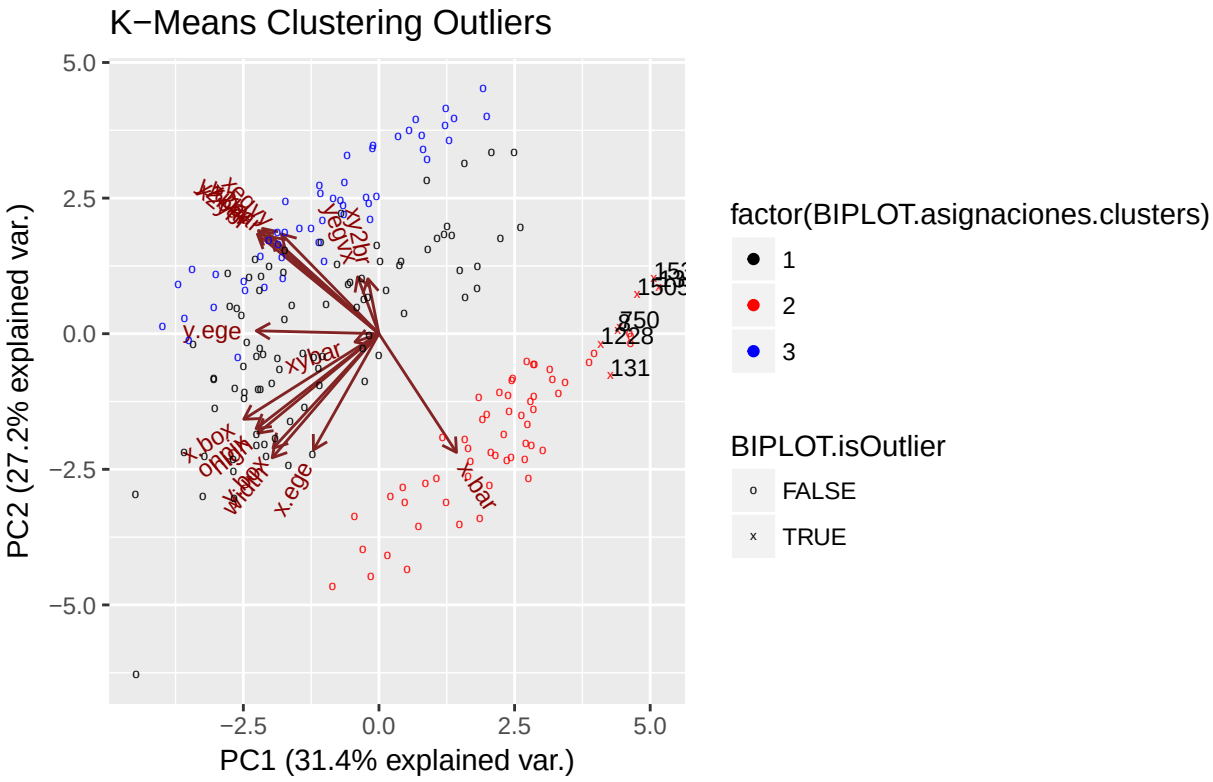
```
medoides.valores.normalizados <- mis.datos.numericos.normalizados[modelo.pam$medoids,]

top.clustering.outliers <-top_clustering_outliers(mis.datos.numericos.normalizados,
                                                  modelo.pam$clustering,
                                                  centroides.valores,
                                                  numero.de.outliers)$indices
top.clustering.outliers
```

```
## [1]  92 180 154 176  15   2 146
```

```
modelo.pam$clustering[top.clustering.outliers]
```

```
##  750 1533 1326 1505  131    8 1228
##    2    2    2    2    2    2    2
```

```
BIPLOT.asignaciones.clusters = modelo.pam$clustering
is.pam.outlier = rep(FALSE, numero.de.datos)
is.pam.outlier[top.clustering.outliers] = TRUE
BIPLOT.isOutlier=is.pam.outlier
MiBiPlot_Clustering_Outliers(mis.datos.numericos, "K-Means Clustering Outliers")
```



Utilizando PAM y vemos como queda claramente separado el cluster 2 frente a 1 y 3 pero estos dos clusters se entremezclan en otra zona.

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

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

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

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

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

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

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

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



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


print("Indices de los top k clustering outliers (k-means, usando distancia relativa)")
```

```
## [1] "Indices de los top k clustering outliers (k-means, usando distancia relativa)"
```

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

```
## [1]  52  21 127  96 119   4 141
```

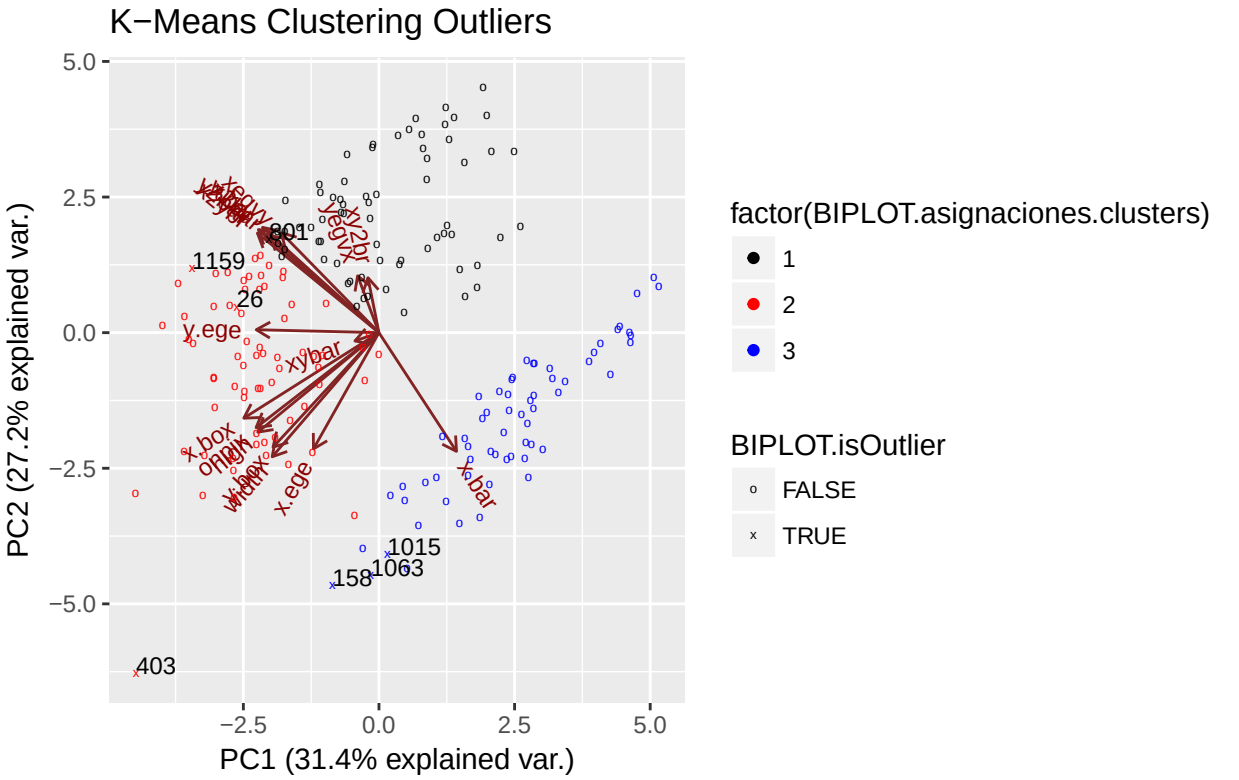
```
print("Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa
```

```
## [1] "Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa
```

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

```
##      403      158     1063      801     1015       26     1159
## 2.436732 1.766269 1.714193 1.679691 1.613713 1.571216 1.563248
```

```
is.outlier = rep(FALSE, numero.de.datos)
is.outlier[top.outliers.kmeans.distancia.relativa$indices] = TRUE
BIPLOT.isOutlier=is.outlier

BIPLOT.cluster.colors        = c("black","red","blue")
BIPLOT.asignaciones.clusters = indices.clustering.LetterRecognition
MiBiPlot_Clustering_Outliers(mis.datos.numericos, "K-Means Clustering Outliers")
```



Vemos como los marcados como outliers en distancia relativa son los que están en los puntos exteriores de cada cluster.

Conclusiones

El manejo de outliers es una parte muy importante a la hora del preprocesamiento de un conjunto de datos, con un correcto tratado podemos lograr mejores resultados de los que tenemos a priori.

Para trabajar con los valores anómalos de un conjunto de datos, es esencial encontrar qué valores lo son. En este trabajo hemos explorado el tratamiento de valores univariantes, valores anómalos dentro de una única característica, y valores multivariantes, que lo son en más de una característica.

Los outliers univariantes los podemos encontrar de dos tipos, normales y extremos. Los calculamos a través del calculo de los cuartiles y de la distancia intercuartil. Será nuestra tarea como analistas de datos el decidir que realizar con ellos.

Con los multivariantes hemos usado distancia de Mahalanobis, Local Outlier Factor (LOF) y k-means. En nuestro caso, utilizando Mahalanobis y usando LOF hemos llegado a los mismos resultados. En el caso de k-means, hemos visto que al tener una gran dispersión en los datos, muchos outliers univariantes, los cluster ocupaban mucha zona aumentando la distancia de los puntos de la frontera del cluster con el centroide y provocando que algún punto quedara lejos de su cluster y muy pegado a otro. Esto se ha solucionado utilizando medias de distancia con los centroides en vez de distancia euclídea. Esta “solución” provoca que clusters muy cercanos vean sus puntos mezclados.

Como vemos, hemos utilizado distintos métodos para detectar anomalías pero ninguno es la panacea, deberemos aplicar uno u otro dependiendo de la distribución de nuestros datos.