Anomalias_D2

Miguel Merelo Hernández

February 11, 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.

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
```

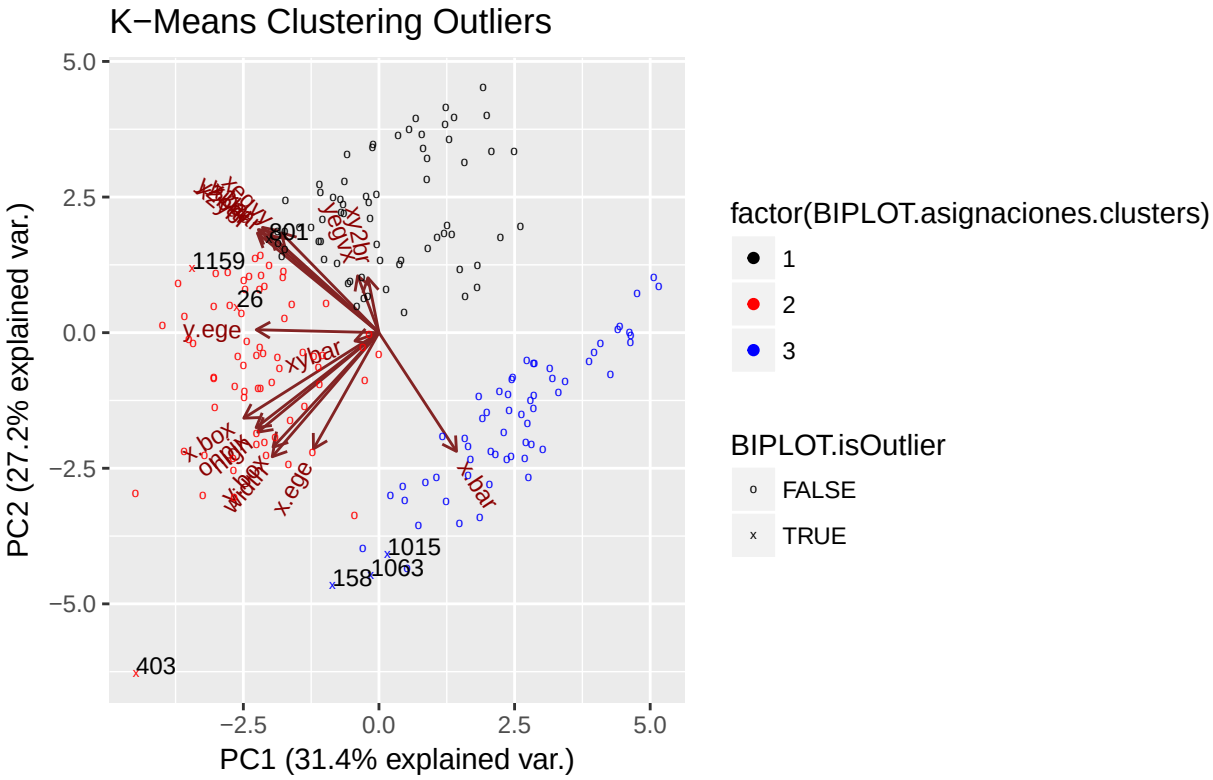
```
## [1]  52  21 127 119  96   4 141
```

```
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, ]
```

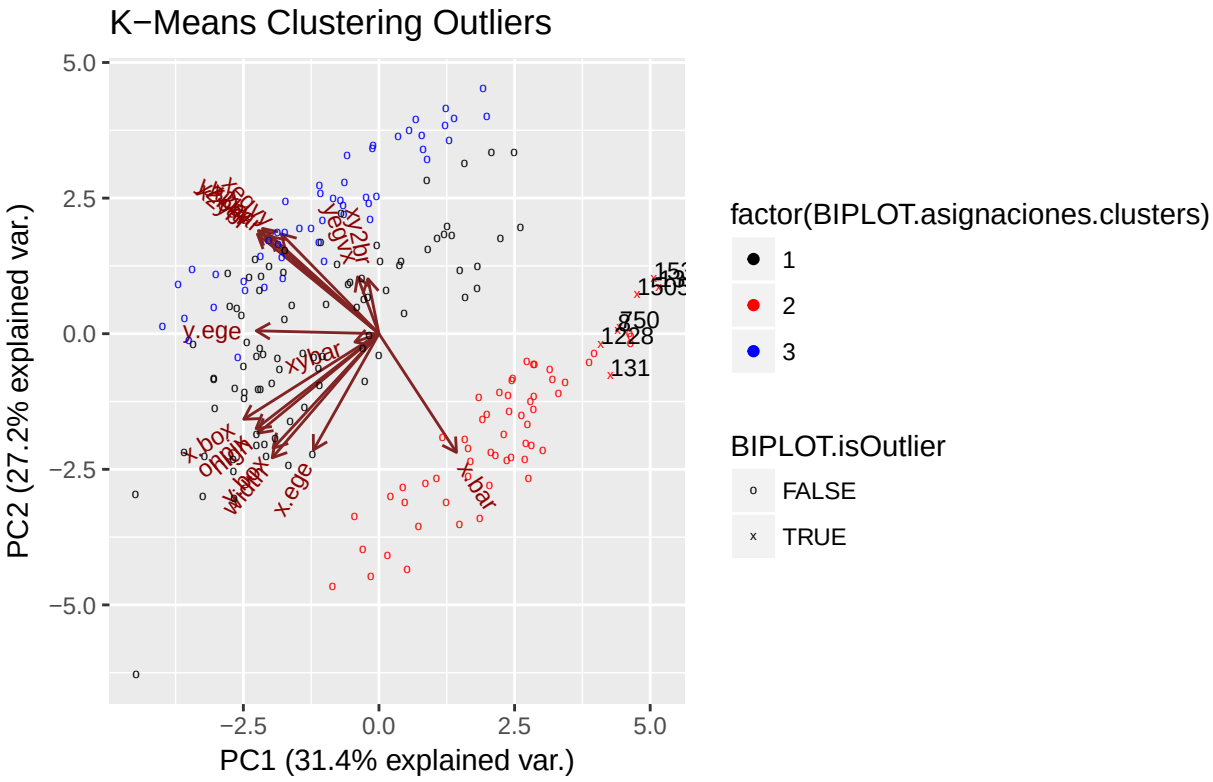
```
medoides.valores.normalizados <- mis.datos.numericos.normalizados[modelo.pam$medoids,]

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
```

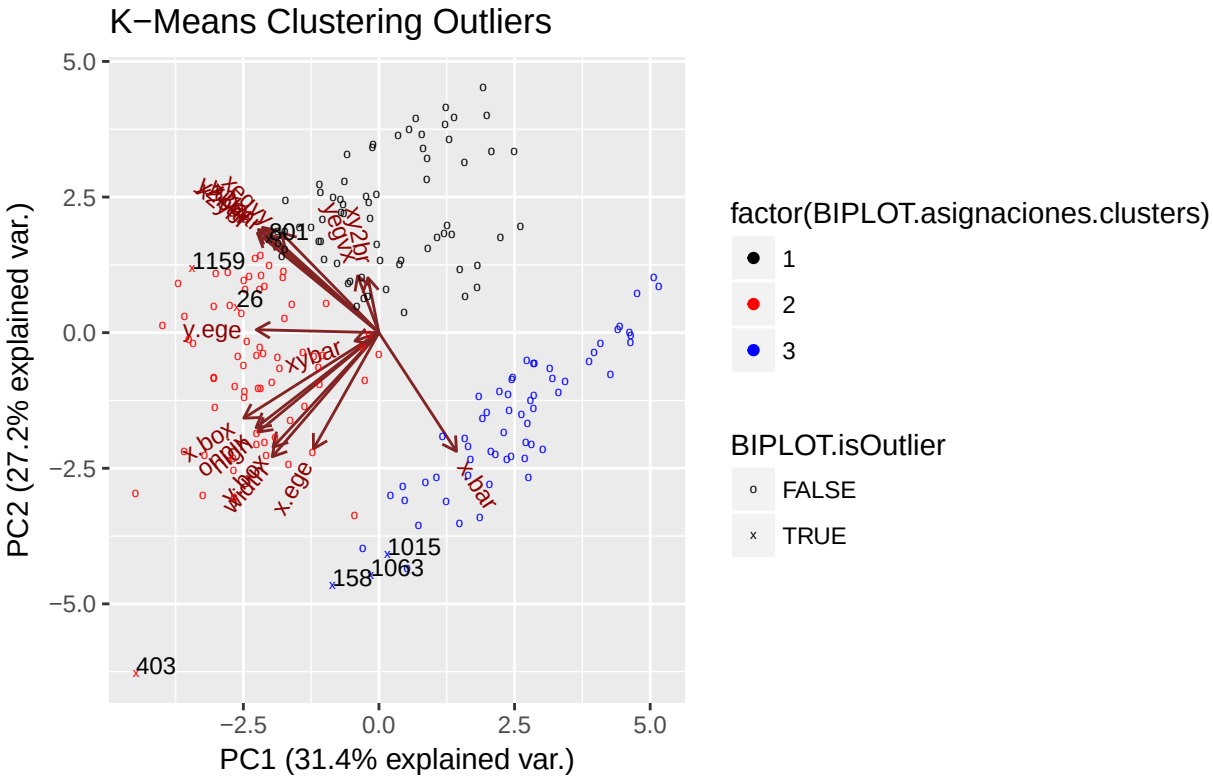
```
print("Distancias a sus centroides de los top k clustering outliers (k-means, usando distancia relativa
```

```
## [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.