PRA2 - Proyecto de visualizacion de datos

Autor: Christopher Irvin Ballon Peralta

Enero 2023

Carga de datos preparados en Practica 1

Se ha usado el archivo de salida de la practica anterior: “clean_data.csv”.

```
# importamos la libreria dplyr
if (!require(dplyr)) install.packages(plyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(dplyr)
# importamos la libreria stringr
if (!require(stringr)) install.packages(stringr)
```

```
## Loading required package: stringr
```

```
library(stringr)
# importamos la libreria plyr
if (!require('plyr')) install.packages('plyr')
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(plyr)
# cargamos el dataset
data <- read.csv("clean_data.csv")
# Seleccionamos una muestra aleatoria del dataset para efectos de no agotar
# memoria del sistema con las funciones utilizadas
data <- sample_frac(data, 0.005)
# Damos formato a la columna productos
data$productos <- str_replace_all(data$productos, "_", "")
# mapeamos los valores dando formato a las variables categoricas
data$segmento <- mapvalues(data$segmento, c("02 - PARTICULARES","03 - UNIVERSITARIO","01 - TOP", "D"), c
(2,3,1,0))
data$nomprov <- mapvalues(data$nomprov, c("CORUÑA, A", "PALMAS, LAS",
                                          "BALEARS, ILLES", "RIOJA, LA"),
                          c("CORUNYA", "PALMAS", "ILLES BALEARS", "LA RIOJA"))
head(data)
```

```
##   ncodpers ind_empleado sexo age antiguedad canal_entrada  nomprov
## 1   191894            N    V  46        182           KAT   MALAGA
## 2  1403947            N    V  39          6           KHM   MADRID
## 3   694775            N    V  40         97           KFC ZARAGOZA
## 4   664275            N    V  40        103           KFC   MADRID
## 5   293552            N    V  43        153           KAT VALENCIA
## 6  1228666            N    H  25         19           KHF  SEVILLA
##   ind_actividad_cliente     renta segmento                productos
## 1                     1 110388.37        2 000010000000101001100111
## 2                     1 115893.36        2 001000000000000000000001
## 3                     0 125194.35        2 001000010000000000000000
## 4                     0  98755.26        2 001000000000000000000000
## 5                     0  56296.26        2 001000000000000000000000
## 6                     0  85915.05        3 001000000000000000000000
```

Descripción de caracteristicas:

ncodpers Codigo del cliente

ind_empleado Indice de empleado: A activo, B ex empleado, F filial, N no empleado, P pasivo

sexo Genero del cliente

age Edad

antiguedad antiguedad del cliente (en meses)

canal_entrada Canal por el cual el cliente fue ingresado

nomprov Nombre de provincia

ind_actividad_cliente Indice de actividad (1, cliente activo; 0, cliente inactivo)

renta Ingreso bruto por hogar

segmento segmentación: 01 - VIP, 02 - Individuos 03 - Graduado universitario

productos codigo binario que indica productos con que cuenta el cliente, en el siguiente orden:

Cuenta de ahorros, Garantia, Cuenta corriente, Cuenta derivada, Cuenta de nomina de sueldos, Cuenta junior, Cuenta Más particular, CUenta particular, Cuenta Particular Plus, Deposito a corto plazo, Deposito a mediano plazo, Deposito a largo plazo, Cuenta electronica, Cuenta de fondos, Prestamo hipotecario, Pensión, Prestamo, Cuenta de impuestos, Tarjeta de credito, Seguro, Cuenta domestica, nomina de sueldo, Pensiones, Debito directo.

---

1 Metodo no supervisado

Seleccionamos las caracteristicas que se utilizaran en la clasificación: (valores numericos)

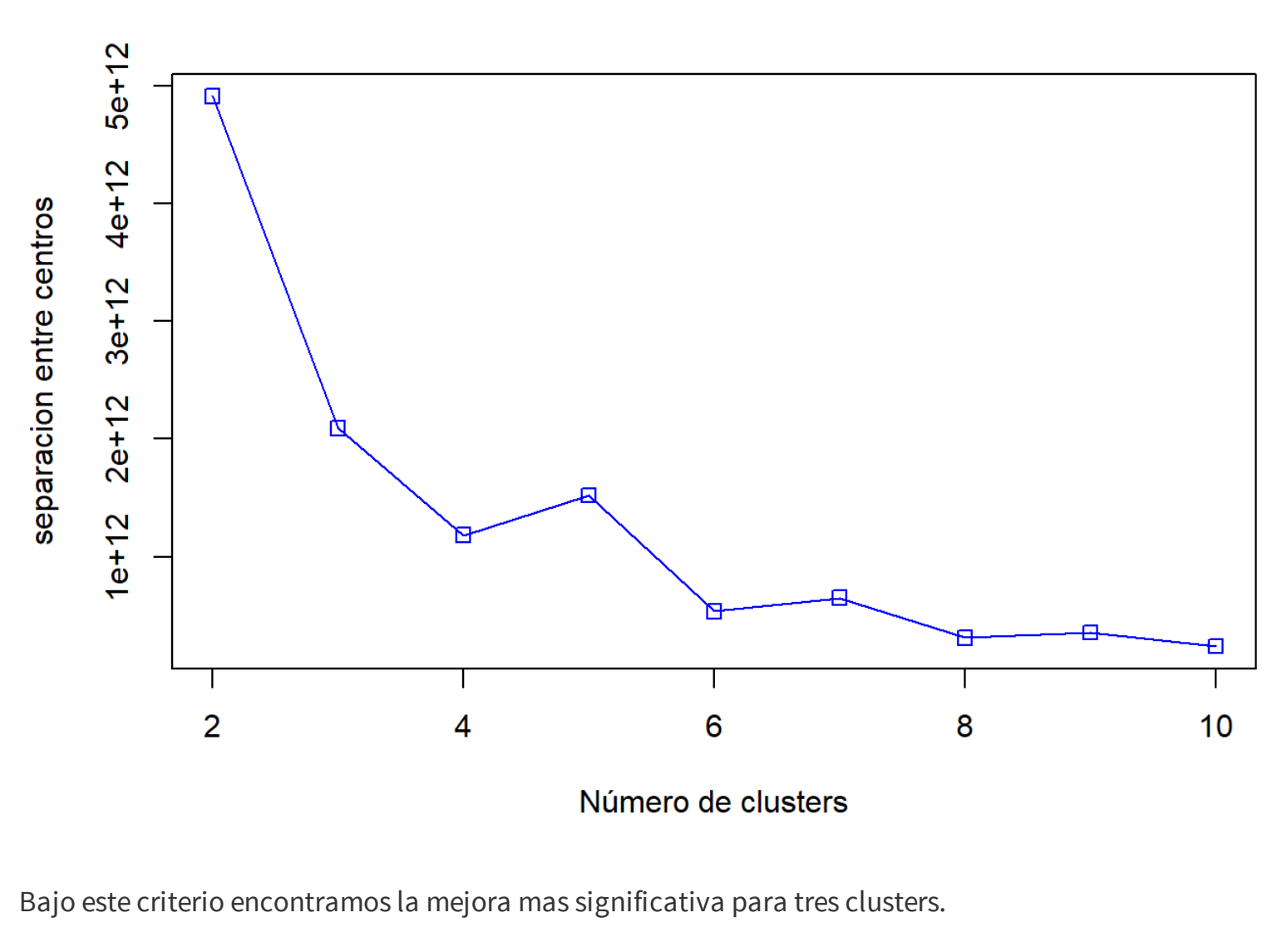
```
clients <- na.omit(data[c(4:5,9:10)])
```

Realizamos un grafico de hombro para encontrar el numero optimo de grupos:

```
#importamos la libreria cluster
if (!require('cluster')) install.packages('cluster')
```

```
## Loading required package: cluster
```

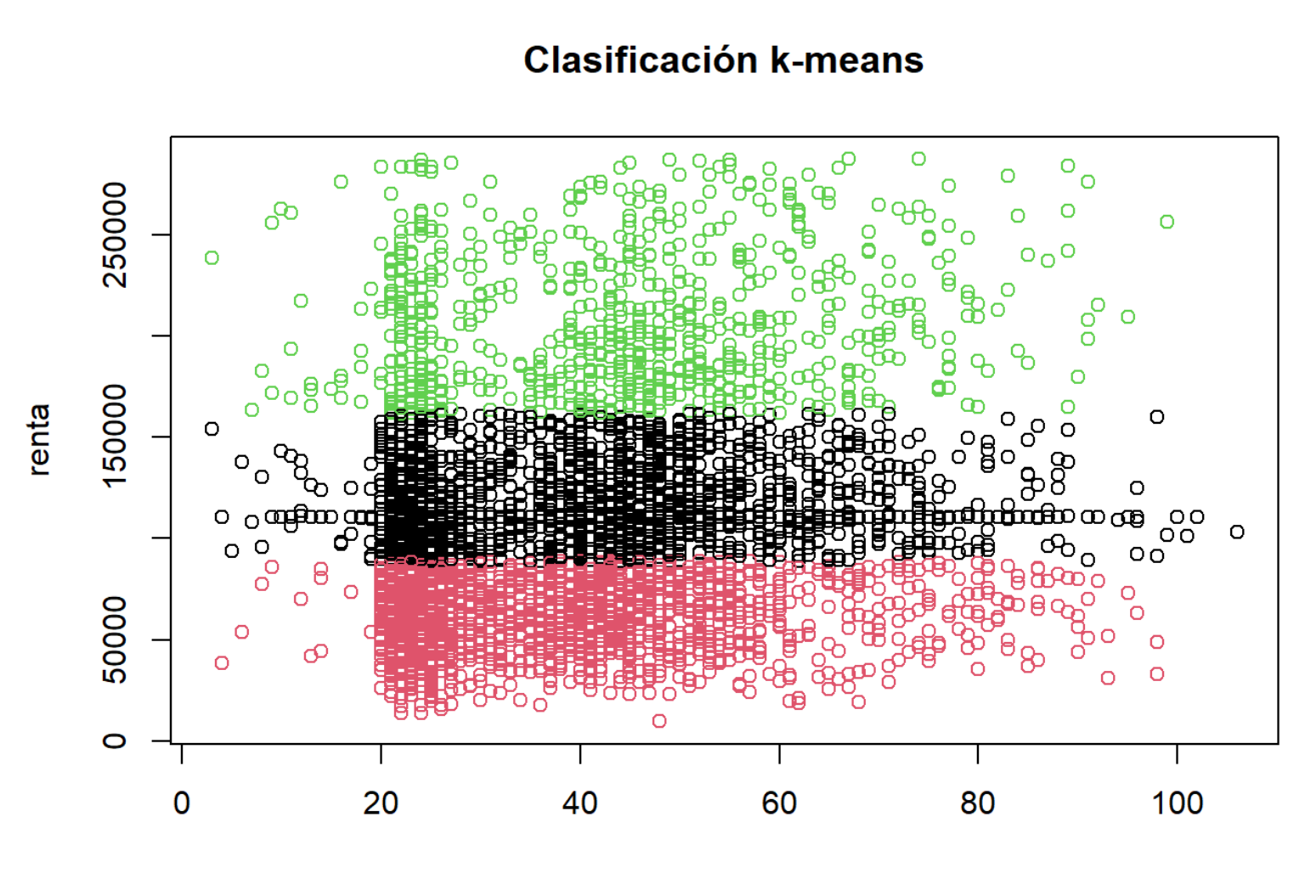
```
library(cluster)
#Realizamos el grafico de hombro
elbow <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(clients, i)
  elbow[i] <- fit$tot.withinss
}
plot(2:10,elbow[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="separacion entre centro
s")
```



Bajo este criterio encontramos la mejora mas significativa para tres clusters.

Ahora utilizamos el algoritmo kmeans e indicamos que se clasifique en 3 grupos:

```
clients3clusters <- kmeans(clients, 3)

# Renta y edad
plot(clients[c(1,3)], col=clients3clusters$cluster, main="Clasificación k-means")
```



Observamos cernania entre los puntos y grupos correctamente segmentados.

Conclusión:

Dado que el dataset fue pensado para ofrecer una campaña de productos financieros, no se tiene en claro el número de grupos que deben distinguirse ni los criterios que se deben utilizar.

Con ello, no podemos validar si la clasificación es correcta al carecer de una varible objetivo.

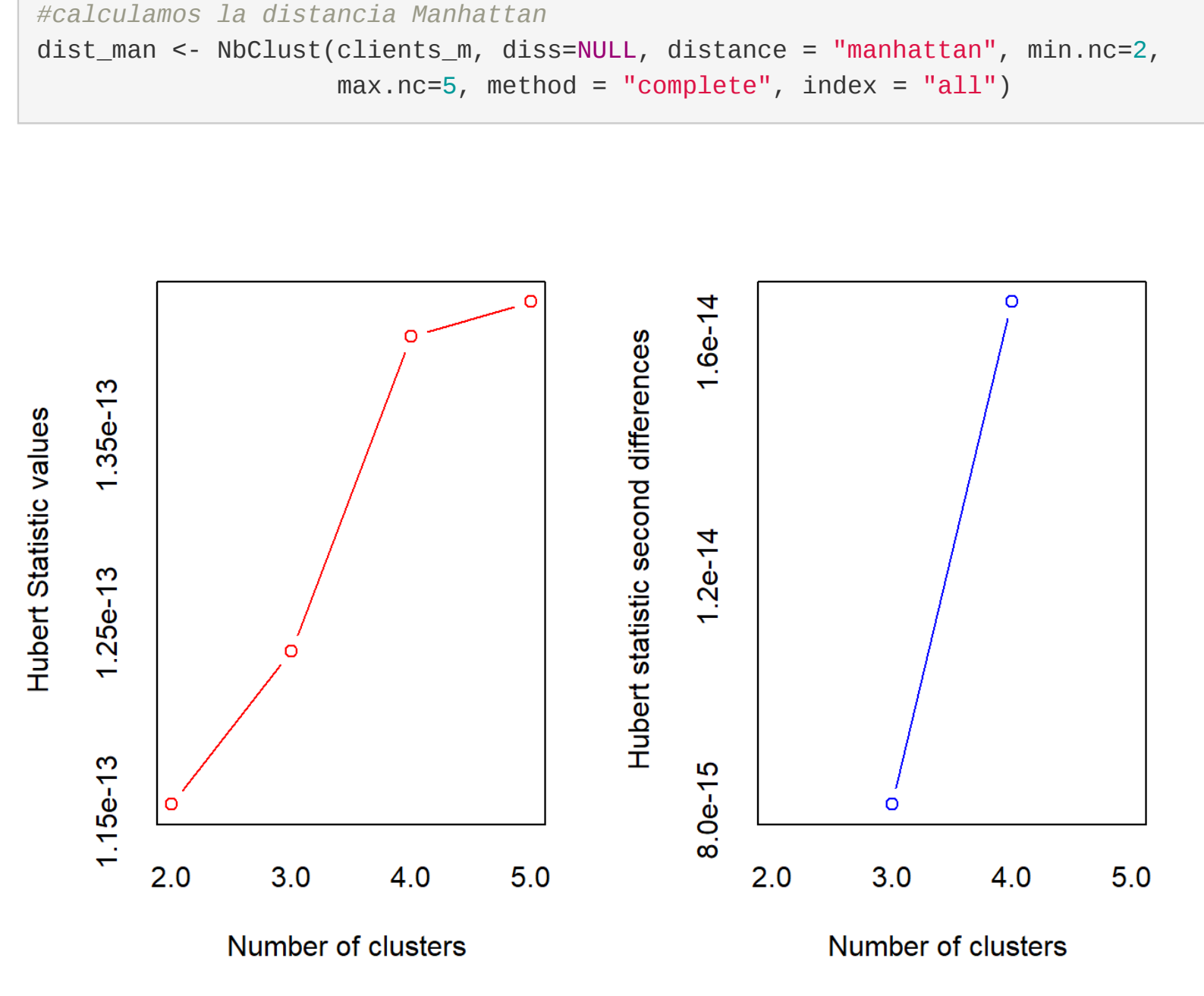
1.1 Metrica Manhattan

El metodo anterior utilizaba la metrica euclidiana para calcular las distancias y así encontrar el número optimo de clusters; ahora planteamos utilizar la **metrica Manhattan** (distancia absoluta).

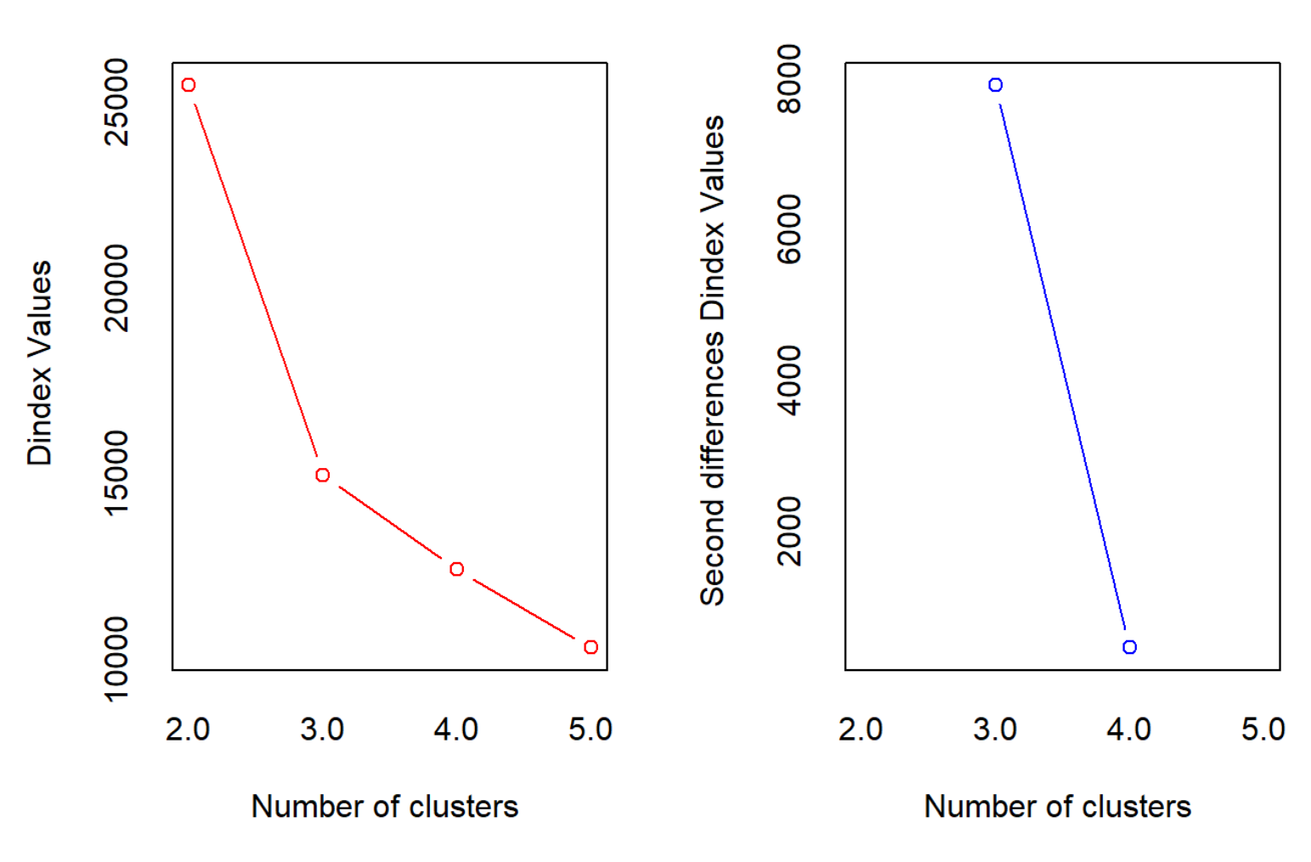
```
# importamos la libreria
if (!require('NbClust')) install.packages('NbClust')
```

```
## Loading required package: NbClust
```

```
library(NbClust)
#convertimos los datos a matriz:
clients_m <- data.matrix(clients)
#calculamos la distancia Manhattan
dist_man <- NbClust(clients_m, diss=NULL, distance = "manhattan", min.nc=2,
                   max.nc=5, method = "complete", index = "all")
```



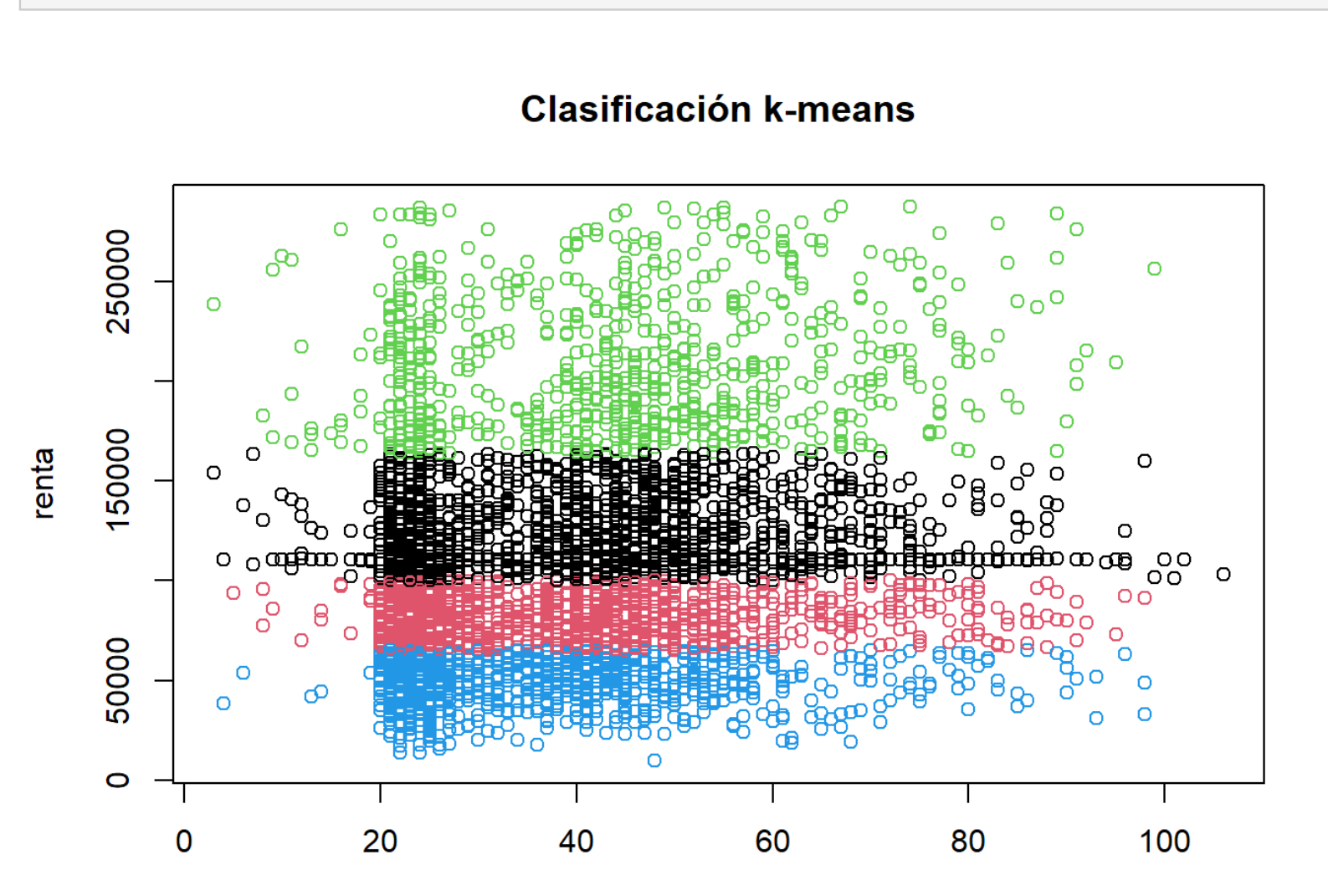
```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 7 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 5 proposed 5 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

Observamos que para esta metrica, el número optimo de clusters es de 4, por ello volvemos a correr el modelo anterior con el nuevo número de clusters:

```
clients3clusters <- kmeans(clients, 4)

# Renta y edad
plot(clients[c(1,3)], col=clients3clusters$cluster, main="Clasificación k-means")
```



Exportamos estos resultados para visualizarlos en Flourish mas adelante.

```
to_plot <- data.frame(clients[c(1,3)], clients3clusters$cluster)
write.csv(to_plot, file="kmeans.csv")
```

Observamos una segmentación correcta para 4 clusters.

Conclusión:

La segunda metrica usada muestra una mejor clasificación que calculando la distancia euclidea (primer caso).

---

2 DBSCAN y OPTICS

Ulizaremos el algoritmo optics para calcular la alcanzabilidad:

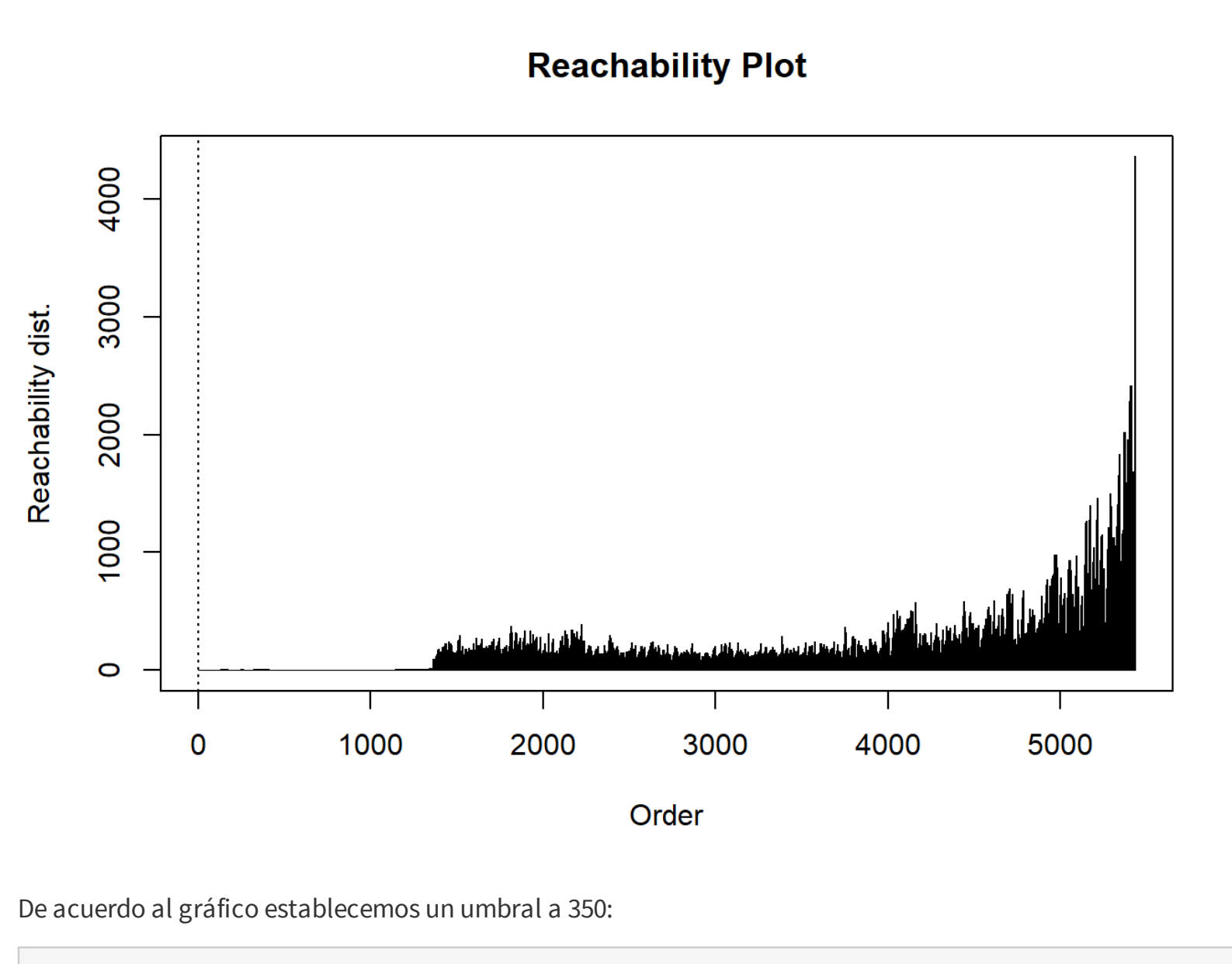
```
#importamos la libreria DBSCAN
if (!require('dbscan')) install.packages('dbscan')
```

```
## Loading required package: dbscan
```

```
##
## Attaching package: 'dbscan'
```

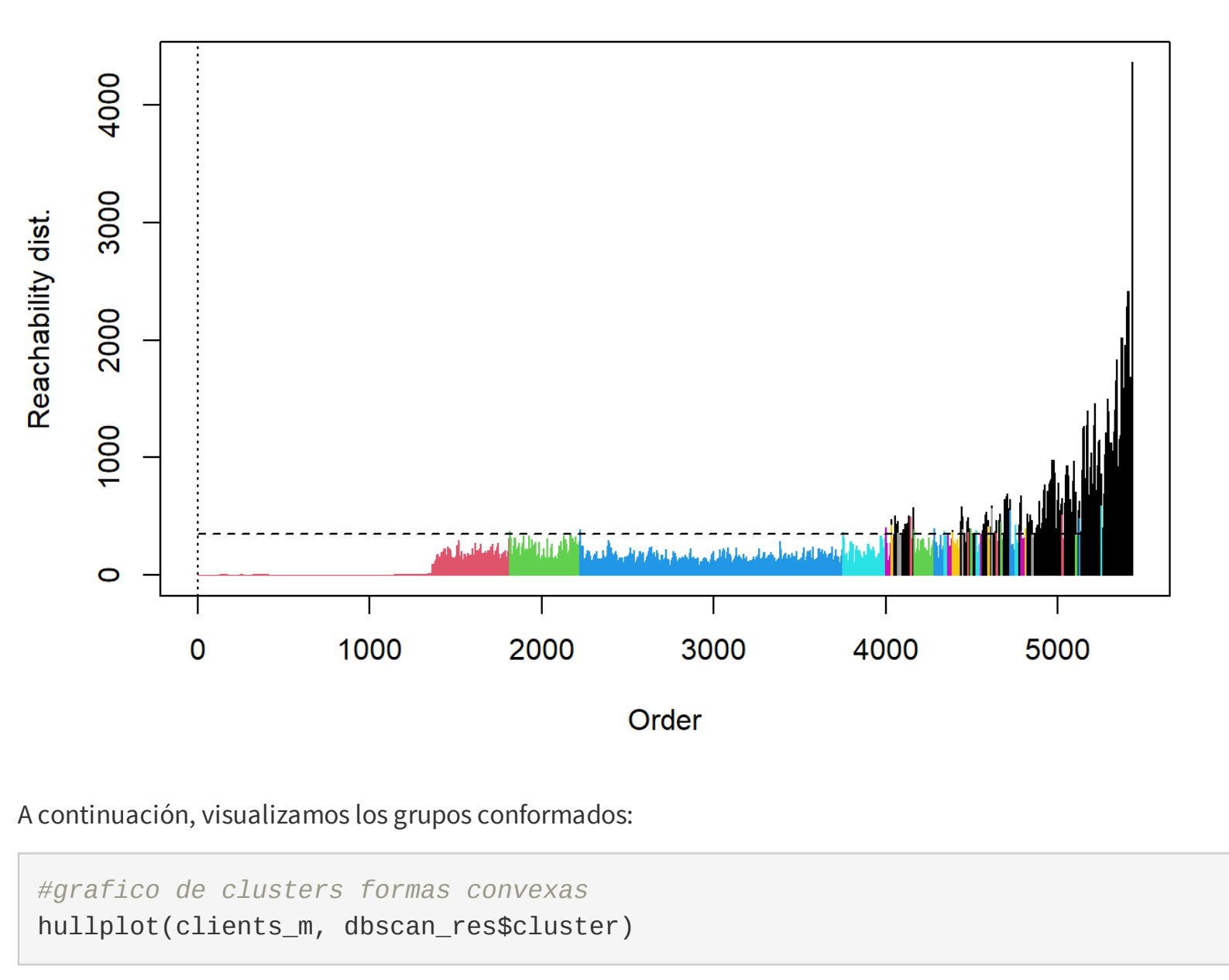
```
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```
library(dbscan)


#Corremos optics en el dataset
optics_res <- optics(clients_m, minPts = 10)

#Gráfico de alcanzabilidad
plot(optics_res)
```



De acuerdo al gráfico establecemos un umbral a 350:

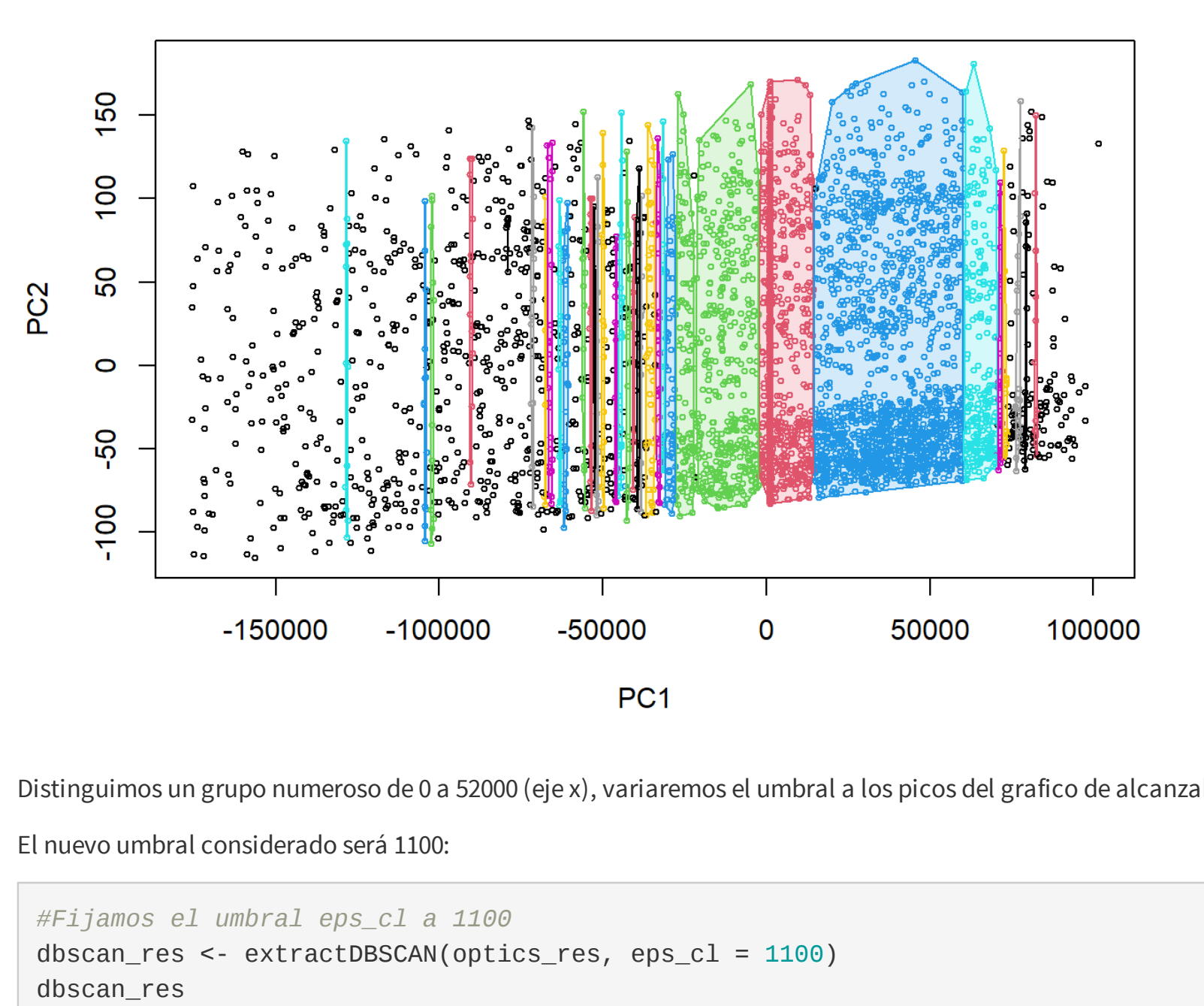
```
#Fijamos el umbral eps_cl a 350
dbscan_res <- extractDBSCAN(optics_res, eps_cl = 350)
plot(dbscan_res)
```



A continuación, visualizamos los grupos conformados:

```
#grafico de clusters formas convexas
hullplot(clients_m, dbscan_res$cluster)
```

```
## Warning in hullplot(clients_m, dbscan_res$cluster): Not enough colors. Some
## colors will be reused.
```



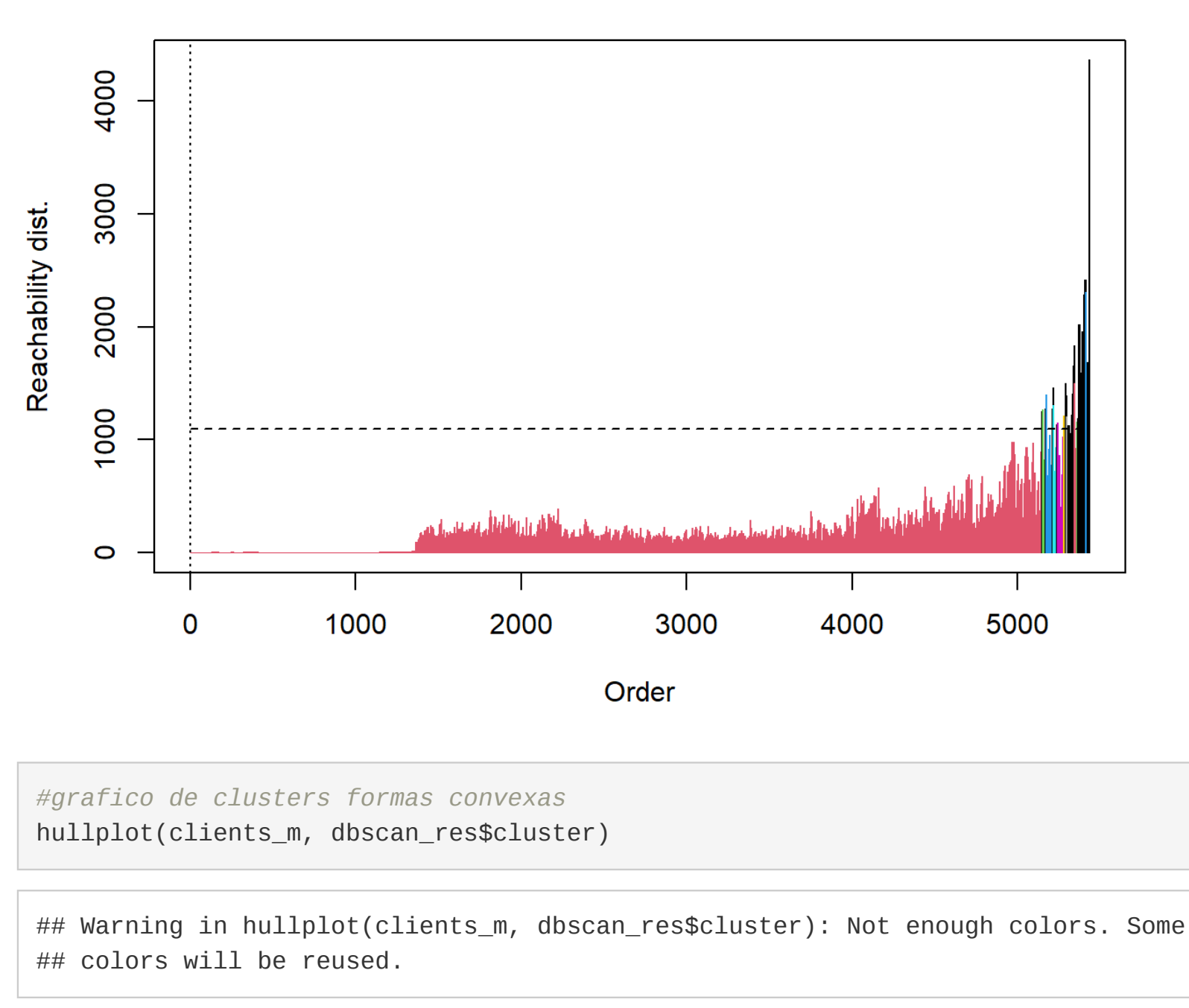
Distinguimos un grupo numeroso de 0 a 52000 (eje x), variaremos el umbral a los picos del grafico de alcanzabilidad del lado derecho.

El nuevo umbral considerado será 1100:

```
#Fijamos el umbral eps_cl a 1100
dbscan_res <- extractDBSCAN(optics_res, eps_cl = 1100)
dbscan_res
```

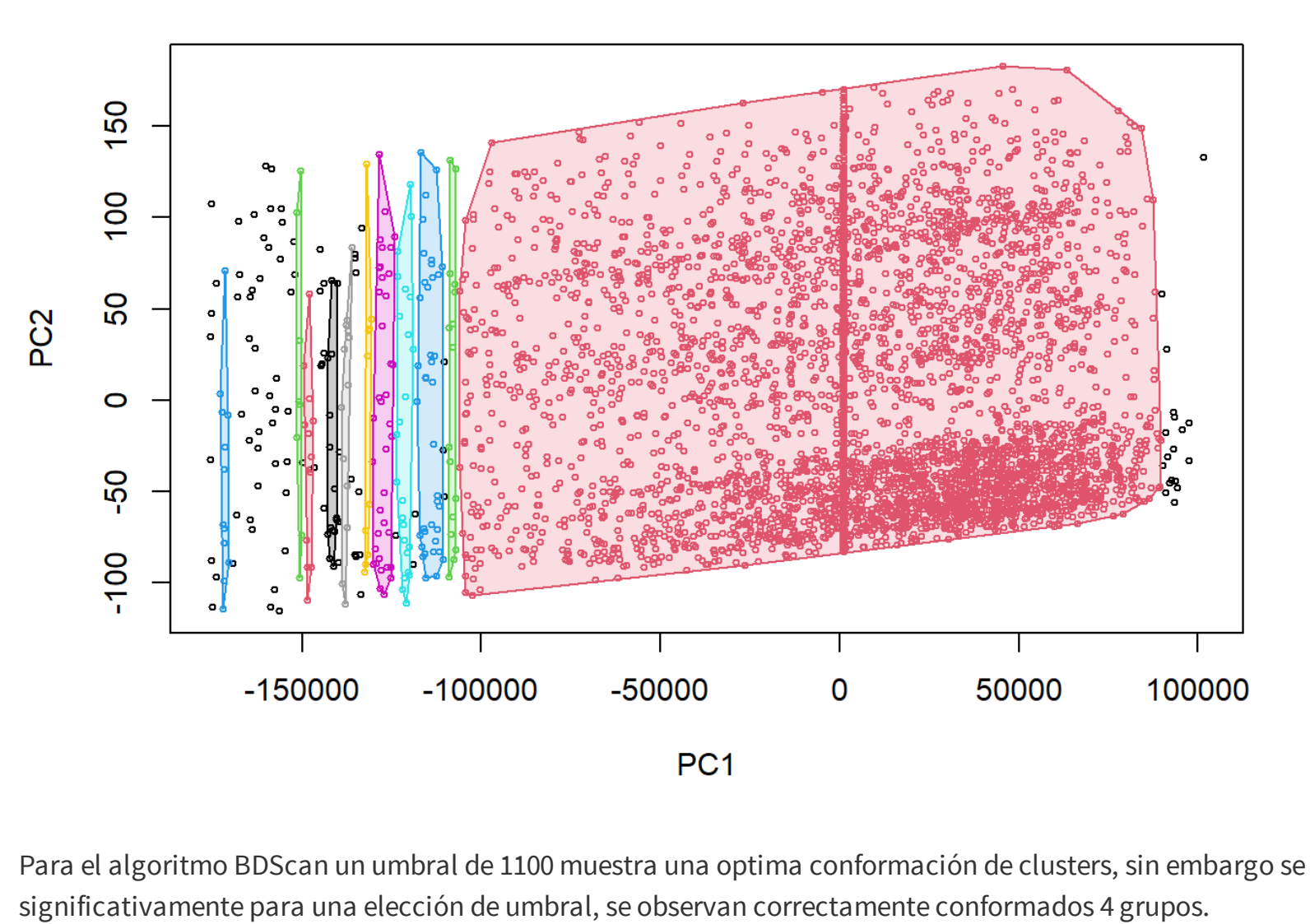
```
## OPTICS ordering/clustering for 5434 objects.
## Parameters: minPts = 10, eps = 8841.05333050876, eps_cl = 1100, xi = NA
## The clustering contains 11 cluster(s) and 97 noise points.
##
##    0    1    2    3    4    5    6    7    8    9   10   11
##   97 5149   16   39   24   38   10   13   15   13    8   12
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

```
plot(dbscan_res)
```



```
#grafico de clusters formas convexas
hullplot(clients_m, dbscan_res$cluster)
```

```
## Warning in hullplot(clients_m, dbscan_res$cluster): Not enough colors. Some
## colors will be reused.
```



Conclusión:

Para el algoritmo BDScan un umbral de 1100 muestra una optima conformación de clusters, sin embargo se observa ruido que influye significativamente para una elección de umbral, se observan correctamente conformados 4 grupos.