hatco-r-analysis

Jordi López Sintas

5 de septiembre de 2014

Paquete estadístico en R para poder leer ficheros de datos en formato SPSS o de otros programas estadísticos:

```
library(foreign)
```

Lectura del fichero de datos en formato SPSS. Recuerda indicar el directorio en el que se encuentra el fichero. Si lo has grabado en el disquet, entonces debrás dar el argumento "a:hatconuevo.sav" en lugar del que tienes en el ejemplo siguiente:

```
#hatco<-read.spss("hatconuevo.sav")
#t(head(hatco))
#leer hatco.completo.csv
hatco <- read.csv("hatco.completo.csv", sep=";", dec=",")

#Mostramos las 6 primeras líneas del fichero de datos
head(hatco)
```

```
##   DELSPEED PRICELEV PRICEFLE MANUFIMA SERVICE SALESFOR PRODUCTQ TAMEMP
## 1      4.1      0.6      6.9      4.7     2.4      2.3      5.2  Small
## 2      1.8      3.0      6.3      6.6     2.5      4.0      8.4  Large
## 3      2.7      1.0      7.1      5.9     1.8      2.3      7.8  Large
## 4      4.6      2.4      9.5      6.6     3.5      4.5      7.6  Small
## 5      2.4      1.6      8.8      4.8     2.0      2.8      5.8  Small
## 6      2.8      1.4      8.1      3.8     2.1      1.4      6.6  Large
##   USAGELEV SATISFLE             ESPCOMPR      ESTRCOMP
## 1       32      4.2 Total Value Analysis Decentralized
## 2       43      4.3 Specification Buying   Centralized
## 3       32      3.9 Specification Buying   Centralized
## 4       46      5.8 Total Value Analysis Decentralized
## 5       32      4.3 Total Value Analysis Decentralized
## 6       39      4.4 Specification Buying   Centralized
##                    INDUSTRI  SITCOMP
## 1 Industry A Classification New Task
## 2          Other Industries New Task
## 3 Industry A Classification New Task
## 4 Industry A Classification New Task
## 5          Other Industries New Task
## 6          Other Industries New Task
```

```
names(hatco)
```

```
##  [1] "DELSPEED" "PRICELEV" "PRICEFLE" "MANUFIMA" "SERVICE"  "SALESFOR"
##  [7] "PRODUCTQ" "TAMEMP"   "USAGELEV" "SATISFLE" "ESPCOMPR" "ESTRCOMP"
## [13] "INDUSTRI" "SITCOMP"
```

```
str(hatco)
```

```
## 'data.frame':    100 obs. of  14 variables:
##  $ DELSPEED: num  4.1 1.8 2.7 4.6 2.4 2.8 3.7 3.4 2.4 2.4 ...
##  $ PRICELEV: num  0.6 3 1 2.4 1.6 1.4 1.5 0.4 1.5 1.5 ...
##  $ PRICEFLE: num  6.9 6.3 7.1 9.5 8.8 8.1 8.6 8.3 6.7 6.6 ...
##  $ MANUFIMA: num  4.7 6.6 5.9 6.6 4.8 3.8 5.7 2.5 4.8 4.8 ...
##  $ SERVICE : num  2.4 2.5 1.8 3.5 2 2.1 2.7 1.2 1.9 1.9 ...
##  $ SALESFOR: num  2.3 4 2.3 4.5 2.8 1.4 3.7 1.7 2.5 2.5 ...
##  $ PRODUCTQ: num  5.2 8.4 7.8 7.6 5.8 6.6 6.7 5.2 7.2 7.2 ...
##  $ TAMEMP  : Factor w/ 2 levels "Large","Small": 2 1 1 2 2 1 2 2 1 1 ...
##  $ USAGELEV: int  32 43 32 46 32 39 38 35 36 36 ...
##  $ SATISFLE: num  4.2 4.3 3.9 5.8 4.3 4.4 5 3.3 3.7 3.7 ...
##  $ ESPCOMPR: Factor w/ 2 levels "Specification Buying",..: 2 1 1 2 2 1 2 2 1 1 ...
##  $ ESTRCOMP: Factor w/ 2 levels "Centralized",..: 2 1 1 2 2 1 2 2 1 1 ...
##  $ INDUSTRI: Factor w/ 2 levels "Industry A Classification",..: 1 2 1 1 2 2 1 2 2 2 ...
##  $ SITCOMP : Factor w/ 3 levels "Modified Rebuy",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Ahora deberás seleccionar las bases de segmentación que utilizarás con los programas de aglomeración jerárquica y partición. En el ejemplo se han seleccionado las primeras 7. Puedes seleccionar éstas o cualesquiera otras, pero deberás, en cualquier caso, argumentar por qué razón escoges esas variables como bases de segmentación.

```
require(dplyr)
```

```
## 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
```

```
bases<-data.frame(hatco[1:7])
names(bases)
```

```
## [1] "DELSPEED" "PRICELEV" "PRICEFLE" "MANUFIMA" "SERVICE"  "SALESFOR"
## [7] "PRODUCTQ"
```

Ahora deberás tomar otra decisión sobre las bases de segmentación. Se trata de decidir si es necesario hacer alguna transformación en ellas antes de utilizar los algoritmos de clasificación. Si no estás seguro o segura de si es necesario puedes hacer el proceso con las variables transformadas y las originales y observar el resultado de la clasificación. Si coincide, bien, si no entonces deberás tomar una decisión al respecto.

Comprobar que las bases están medidas en la misma escala. Si no lo estuvieran, entonces normalizar las bases de segmentación utilizando la función scale() o bases.norm<-scale(bases)

Si quieres conocer las correlaciones, sólo tienes que utilizar el siguiente comando:

```
cor(bases)
```

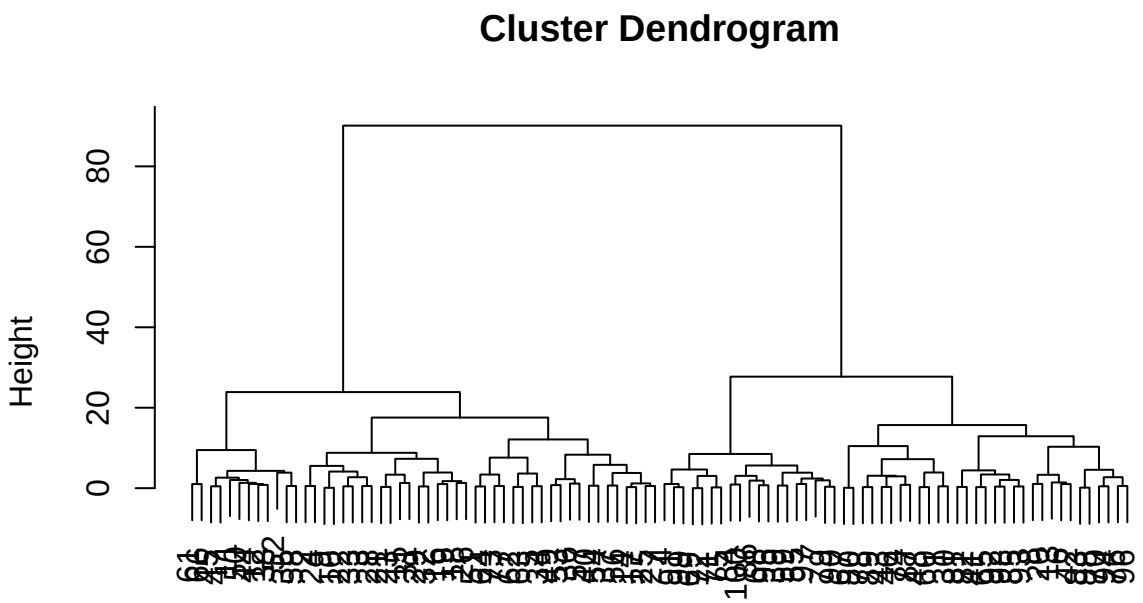
```
##          DELSPEED PRICELEV PRICEFLE MANUFIMA  SERVICE SALESFOR PRODUCTQ
## DELSPEED  1.00000  -0.3492  0.50930  0.05041  0.61190  0.07712 -0.48263
## PRICELEV -0.34923   1.0000 -0.48721  0.27219  0.51298  0.18624  0.46975
## PRICEFLE  0.50930  -0.4872  1.00000 -0.11610  0.06662 -0.03432 -0.44811
## MANUFIMA  0.05041   0.2722 -0.11610  1.00000  0.29868  0.78822  0.19998
## SERVICE   0.61190   0.5130  0.06662  0.29868  1.00000  0.24081 -0.05516
## SALESFOR  0.07712   0.1862 -0.03432  0.78822  0.24081  1.00000  0.17729
## PRODUCTQ -0.48263   0.4697 -0.44811  0.19998 -0.05516  0.17729  1.00000
```

Agrupamos a los clientes y mostramos el resultado de la agrupación

```
bases.hclust<-hclust(dist(bases, method="euclidean"), method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
#Mostramos el resultado de la agrupación
plot(bases.hclust)
```



Ahora calculamos los centros de los grupos formados durante el proceso de agrupación jerárquica y los mostramos.

```
centros.bases<-tapply(as.matrix(bases), list(rep(cutree(bases.hclust, 2), ncol(as.matrix(bases))),col(as
#Visualizamos el resultado
centros.bases
```

```
##      1     2     3     4     5    6     7
## 1 4.46 1.576 8.900 4.926 2.992 2.51 5.904
## 2 2.57 3.152 6.888 5.570 2.840 2.82 8.038
```

Dividimos la muestra con kmeans

```
bases.kmeans2<-kmeans(bases, centros.bases)
```

Para caracterizar a los segmentos utilizamos las medias de las variables originales en los segmentos formados.

```
names(bases.kmeans2)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

El objeto centers contiene la información que buscamos si hemos segmentado con las variables originales

```
bases.kmeans2$centers
```

```
##   DELSPEED PRICELEV PRICEFLE MANUFIMA SERVICE SALESFOR PRODUCTQ
## 1    4.383    1.581    8.900    4.925   2.958    2.525    5.904
## 2    2.575    3.213    6.804    5.598   2.871    2.817    8.127
```

Identificamos a los componentes de los grupos. Para ello utilizamos las variables descriptoras de los segmentos

```
table(hatco$TAMEMP, bases.kmeans2$cluster)
```

```
##
##          1  2
##   Large  2 38
##   Small 50 10
```

Como USAGELEV es una variable metrica podemos calcular las medias con la función t.test() o con la función lm()

```
t.test(hatco$USAGELEV ~ bases.kmeans2$cluster)
```

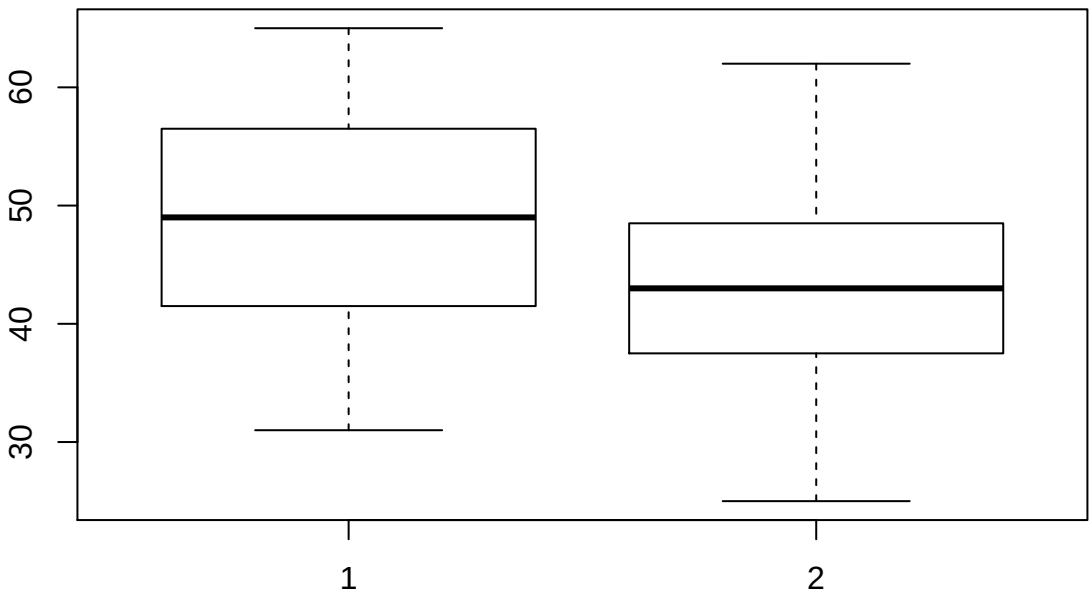
```
##
##  Welch Two Sample t-test
##
## data:  hatco$USAGELEV by bases.kmeans2$cluster
## t = 3.87, df = 97.39, p-value = 0.0001966
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  3.158 9.806
## sample estimates:
## mean in group 1 mean in group 2
##           49.21           42.73
```

```r
summary(lm(hatco$USAGELEV ~ bases.kmeans2$cluster))
```

```
##
## Call:
## lm(formula = hatco$USAGELEV ~ bases.kmeans2$cluster)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.212  -5.979   0.271   6.271  19.271
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                55.69       2.63   21.15  < 2e-16 ***
## bases.kmeans2$cluster      -6.48       1.69   -3.85  0.00021 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.42 on 98 degrees of freedom
## Multiple R-squared:  0.131,  Adjusted R-squared:  0.122
## F-statistic: 14.8 on 1 and 98 DF,  p-value: 0.000214
```

Y ver las medias en un gráfico como el siguiente

```r
boxplot(hatco$USAGELEV ~ bases.kmeans2$cluster)
```



Para las variables categóricas utilizamos la función table()

```r
table(hatco$ESPCOMPR, bases.kmeans2$cluster)
```

```
##
##                          1  2
##   Specification Buying   2 38
##   Total Value Analysis  50 10
```

```
table(hatco$ESTRCOMP, bases.kmeans2$cluster)
```

```
##
##                   1  2
##   Centralized     2 48
##   Decentralized 50  0
```

```
table(hatco$INDUSTRI, bases.kmeans2$cluster)
```

```
##
##                               1  2
##   Industry A Classification 24 26
##   Other Industries          28 22
```

```
table(hatco$SITCOMP, bases.kmeans2$cluster)
```

```
##
##                    1  2
##   Modified Rebuy 10 22
##   New Task       10 24
##   Straight Rebuy 32  2
```

Ahora comprobariamos si la segmentación obtenida con las bases originales varía cuando las sustituimos por sus componentes principales

Comprobar que la correlación entre las bases de segmentación no suponga un problema para la segmentacion. Visualizamos las correlaciones entre las bases de segmentación

```
cor(bases)
```

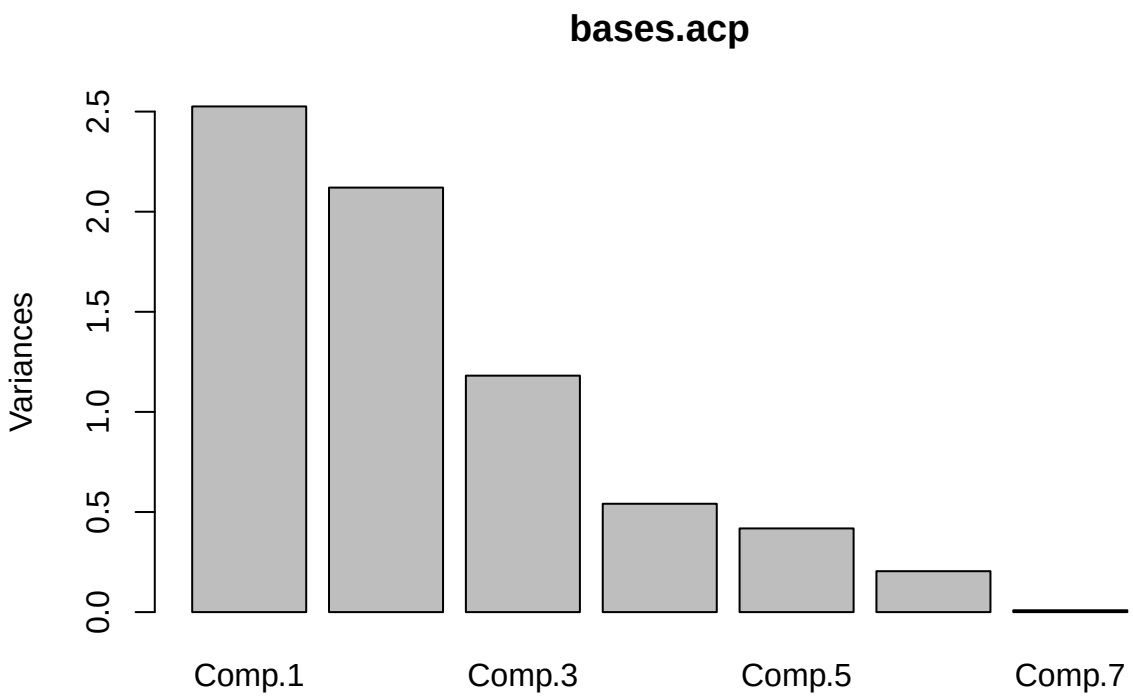
```
##          DELSPEED PRICELEV PRICEFLE MANUFIMA  SERVICE SALESFOR PRODUCTQ
## DELSPEED  1.00000  -0.3492  0.50930  0.05041  0.61190  0.07712 -0.48263
## PRICELEV -0.34923   1.0000 -0.48721  0.27219  0.51298  0.18624  0.46975
## PRICEFLE  0.50930  -0.4872  1.00000 -0.11610  0.06662 -0.03432 -0.44811
## MANUFIMA  0.05041   0.2722 -0.11610  1.00000  0.29868  0.78822  0.19998
## SERVICE   0.61190   0.5130  0.06662  0.29868  1.00000  0.24081 -0.05516
## SALESFOR  0.07712   0.1862 -0.03432  0.78822  0.24081  1.00000  0.17729
## PRODUCTQ -0.48263   0.4697 -0.44811  0.19998 -0.05516  0.17729  1.00000
```

Si la correlación es elevada, transformamos las bases en unas nuevas variables llamadas componentes principales. Ahora vamos a calcular los componentes principales para comprobar si el resutlado cambia

```
bases.acp<-princomp(bases, cor=T)
```

Podemos visualizar la varianza explicada por cada componente principal. Al utilizar las correlación (acotadas entre -1 y 1) la variacion explicada por los componentes principales es igual al número de variables originales y la varianza explicada por cada componentes estará entre 0 y 7

```
plot(bases.acp)
```




```
summary(bases.acp)
```

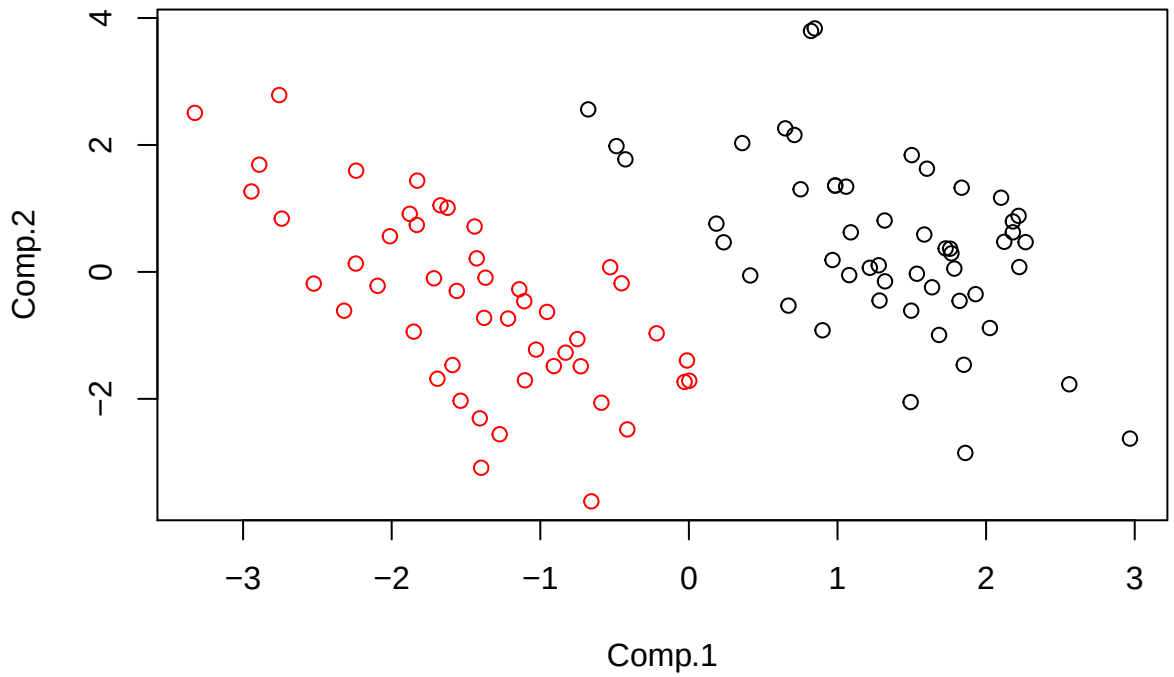
```
## Importance of components:
##                         Comp.1 Comp.2 Comp.3  Comp.4  Comp.5 Comp.6
## Standard deviation     1.5893 1.4562 1.0868 0.73564 0.64655 0.4521
## Proportion of Variance 0.3608 0.3029 0.1687 0.07731 0.05972 0.0292
## Cumulative Proportion  0.3608 0.6637 0.8325 0.90977 0.96949 0.9987
##                           Comp.7
## Standard deviation     0.095713
## Proportion of Variance 0.001309
## Cumulative Proportion  1.000000
```

Vemos que con cuatro cuatro componentes explicamos el 90% de la variación. Ahora para segmentar utilizamos la puntuación de los clientes en los nuevos componentes principales Esa puntuación está recogida en el objeto scores de la lista bases.acp La asignamos al objeto bases.puntos

```
bases.puntos<-bases.acp$scores
```

Si queremos visualizar el resultado de la clasificacion que hemos realizado antes, podemos utilizar los componentes de esta forma:

```
plot(bases.puntos[,1:2], col=bases.kmeans2$cluster)
```

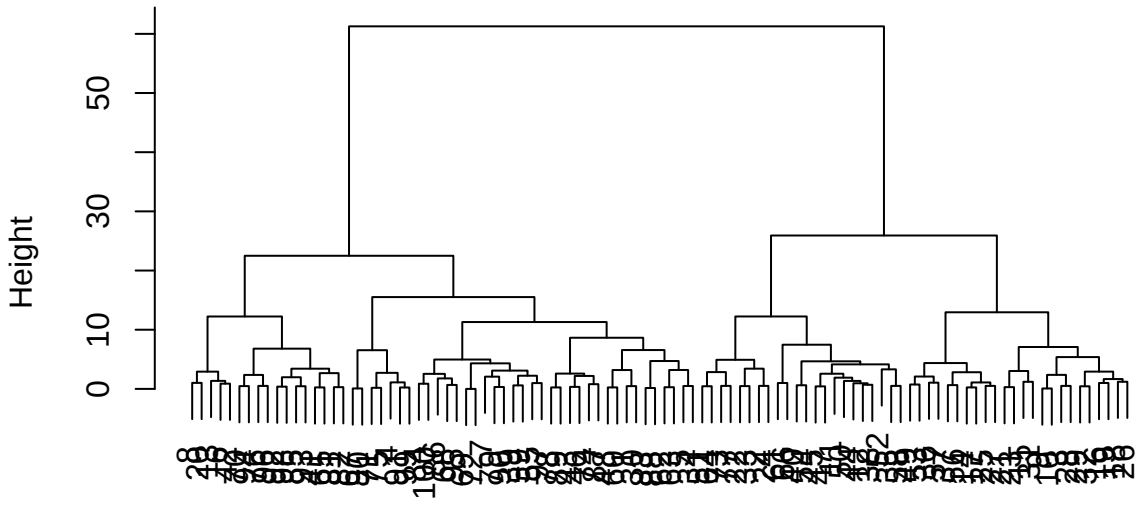


Ahora volvemos a realizar el proceso de segmentación, pero con los componentes principales. Primero la agrupación jerárquica

```
bases.puntos.hclust<-hclust(dist(bases.puntos), method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
#visualizamos la heterogeneidad y la comparamos con el la mostrada con los datos originales
plot(bases.puntos.hclust)
```

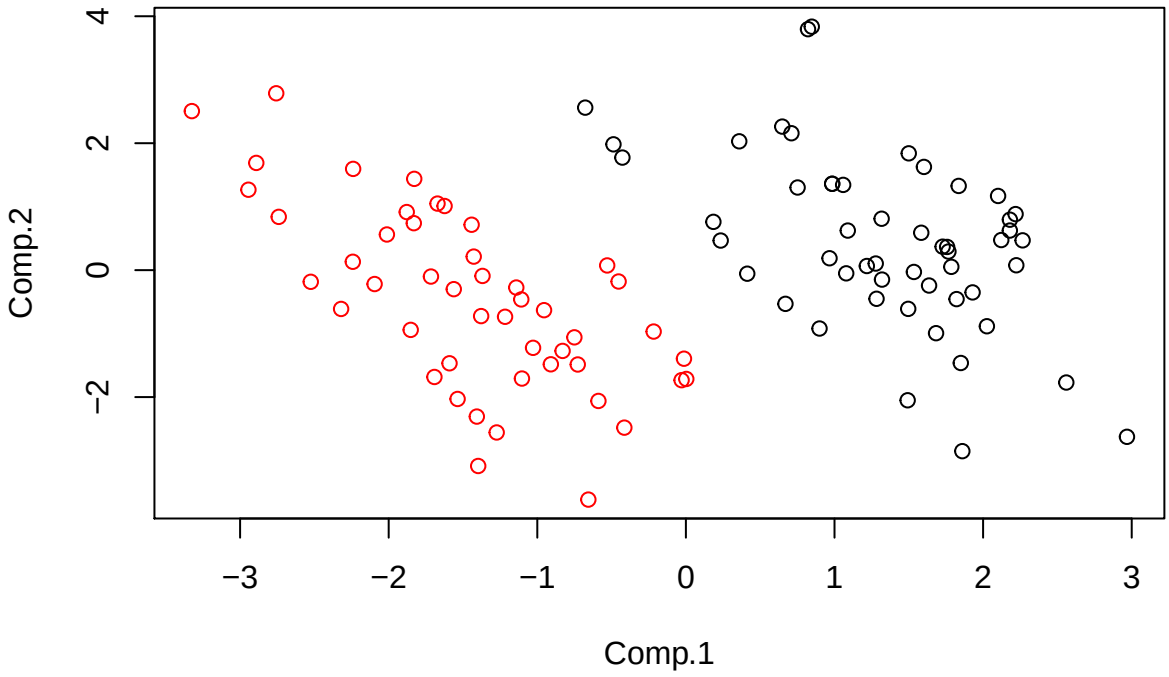



```
#Calculamos los centros de los grupos
centros.bases.puntos<-tapply(bases.puntos, list(rep(cutree(bases.puntos.hclust, 2), ncol(bases.puntos))
#isualizamos los centros
centros.bases.puntos
```

```
##        1       2        3        4        5        6         7
## 1  1.264  0.4286  0.06955  0.03731 -0.04071  0.01005  0.002706
## 2 -1.484 -0.5031 -0.08165 -0.04380  0.04779 -0.01180 -0.003176
```

De nuevo partimos la muestra con kmeans

```
bases.puntos.kmeans2<-kmeans(bases.puntos, centros.bases.puntos)
plot(bases.puntos[,1:2], col=bases.puntos.kmeans2$cluster)
```



Y

Comparamos el resultado

```
table(bases.kmeans2$cluster, bases.puntos.kmeans2$cluster)
```

```
##
##      1  2
##   1 52  0
##   2  0 48
```

```
bases.puntos.kmeans2
```

```
## K-means clustering with 2 clusters of sizes 52, 48
##
## Cluster means:
##   Comp.1  Comp.2   Comp.3   Comp.4    Comp.5   Comp.6    Comp.7
## 1  1.331  0.4471  0.05633  0.03763 -0.005476  0.02485  0.002257
## 2 -1.442 -0.4844 -0.06103 -0.04077  0.005933 -0.02692 -0.002445
##
## Clustering vector:
##   [1] 1 2 2 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2
##  [36] 2 2 2 1 2 1 1 1 2 2 1 2 1 2 2 2 2 2 2 1 2 2 2 1 1 2 2 2 2 1 2 1 1 1 1
##  [71] 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 251.3 234.5
##  (between_SS / total_SS =  30.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
options(digits=3)
#Mostramos el valor medio de las bases de segmentación en los grupos
t(aggregate(bases, list(Segmento = bases.puntos.kmeans2$cluster), mean))
```

```
##          [,1] [,2]
## Segmento 1.00 2.00
## DELSPEED 4.38 2.58
## PRICELEV 1.58 3.21
## PRICEFLE 8.90 6.80
## MANUFIMA 4.92 5.60
## SERVICE  2.96 2.87
## SALESFOR 2.52 2.82
## PRODUCTQ 5.90 8.13
```