

Caso Hatco: Análisis y toma de decisiones en el entorno R

Jordi López Sintas

5 de septiembre de 2014

Para introducir en el entorno R los datos recolectados por la dirección de Hatco tenemos varias opciones que veremos a lo largo del curso. Hoy vamos a ver cómo leer ficheros de datos generados por el programa SPSS. En el entorno R de análisis de datos disponemos del paquete **foreign** con el que podemos cómodamente acceder a datos en el formato utilizado por SPSS. Para utilizar el paquete **foreign** tenemos que cargarlo la la función `library()` como mostramos a continuación:

```
options(width=58)
library(foreign)
```

Una vez cargado el paquete podemos utilizar las funciones que los ofrece. Concretamente nos interesa la función `read.spss(<nombre.del.fichero.de.datos>)`. Recuerda indicar el directorio en el que se encuentra el fichero. Si lo has grabado en una memoria usb, entonces en el argumento deberás especificar la dirección completa de la localización del fichero en tu sistema informático. En el ejemplo, el fichero está en el mismo directorio que este documento:

```
hatco<-read.spss("hatconuevo.sav")
#Después de leer los datos podemos listar los datos. En el ejemplo que sigue hemos
#transpuesto la salida ordinaria con el objeto de facilitar la lectura
t(head(hatco))
```

```
##      DELSPEED  PRICELEV  PRICEFLE  MANUFIMA
## [1,] Numeric,100 Numeric,100 Numeric,100 Numeric,100
##      SERVICE    SALESFOR
## [1,] Numeric,100 Numeric,100
```

```
#Aquí mostramos cómo leer el fichero "hatco.completo.csv" en el formato de intercambio csv.
hatco.completo <- read.csv("hatco.completo.csv", sep=";", dec=",")
#Si sólo queremos comprobar que hemos leído los datos correctamente, podemos
#utilizar la función head(<nombre.del.fichero>) que nos mostrará las primeras 6
#líneas del fichero. La función tail() muestra las últimas 6 líneas.
head(hatco)
```

```
## $DELSPEED
##      [1] 4.1 1.8 2.7 4.6 2.4 2.8 3.7 3.4 2.4 2.4 3.0 3.4 2.4
##      [14] 1.8 0.0 2.4 1.9 2.0 2.8 3.8 1.1 3.0 1.0 1.6 2.3 2.5
##      [27] 2.1 2.9 1.9 4.0 0.6 2.0 3.1 2.5 3.4 1.9 1.3 4.0 3.9
##      [40] 3.2 4.9 3.3 5.1 4.1 3.6 5.0 3.4 3.7 2.6 4.5 2.9 4.9
##      [53] 2.3 3.0 3.7 1.6 2.3 3.6 4.2 3.8 3.4 2.6 2.4 3.0 4.8
##      [66] 3.1 6.0 5.5 4.7 3.4 5.3 4.7 3.0 4.6 5.2 3.5 2.8 5.2
##      [79] 4.0 5.9 4.9 5.0 3.1 5.8 5.4 5.4 4.3 3.1 5.1 4.1 4.2
##      [92] 5.3 5.6 3.6 5.2 3.3 4.5 5.5 4.3 6.1
##
## $PRICELEV
```

```

##      [1] 0.6 3.0 1.0 2.4 1.6 1.4 1.5 0.4 1.5 1.5 3.2 3.7 1.0
##     [14] 3.3 2.1 2.0 3.4 2.6 2.4 0.8 2.0 2.0 1.9 2.8 3.7 3.1
##     [27] 3.5 1.2 2.7 0.5 1.6 2.8 2.2 1.8 5.2 3.3 4.2 3.5 2.2
##     [40] 4.1 1.8 0.9 1.4 3.7 4.0 1.3 3.9 0.7 4.8 4.1 2.6 4.4
##     [53] 4.5 3.8 1.4 4.5 3.7 5.4 2.4 0.8 4.6 3.0 2.9 2.8 1.7
##     [66] 4.2 0.9 1.6 1.3 2.0 1.4 1.3 4.0 2.1 1.3 2.8 3.8 2.0
##     [79] 0.9 0.9 2.3 2.5 1.9 0.2 2.1 2.5 1.8 1.9 1.9 1.1 2.5
##     [92] 1.7 2.2 2.2 1.3 2.6 1.6 1.8 2.5 0.5
##
## $PRICEFLE
##      [1] 6.9 6.3 7.1 9.5 8.8 8.1 8.6 8.3 6.7 6.6
##     [11] 6.0 6.4 7.7 7.5 6.9 6.4 7.6 6.5 6.7 8.7
##     [21] 7.2 6.6 7.1 6.1 7.6 7.0 7.4 7.3 5.0 6.7
##     [31] 6.4 5.2 6.7 9.0 5.7 7.9 6.2 6.5 9.1 5.7
##     [41] 7.7 8.6 8.7 5.9 5.8 8.6 5.6 8.2 8.2 6.3
##     [51] 7.7 7.4 8.0 5.5 9.0 6.4 8.3 5.9 9.4 8.3
##     [61] 5.5 8.5 8.4 7.8 7.6 5.1 9.6 9.4 9.9 9.7
##     [71] 9.7 9.9 9.1 7.9 9.7 9.9 8.9 9.3 9.1 9.6
##     [81] 9.3 9.4 10.0 8.8 8.0 9.6 7.6 9.9 9.2 9.3
##     [91] 9.2 8.5 8.2 9.9 9.1 9.7 8.7 8.7 9.3 9.2
##
## $MANUFIMA
##      [1] 4.7 6.6 5.9 6.6 4.8 3.8 5.7 2.5 4.8 4.8 5.3 5.7 3.4
##     [14] 4.5 5.4 4.5 4.6 3.7 4.9 2.9 4.7 6.6 4.5 6.4 5.0 4.2
##     [27] 4.8 6.1 4.9 4.5 5.0 5.0 6.8 5.0 6.0 4.8 5.1 6.0 4.6
##     [40] 5.1 4.3 4.0 4.8 5.5 5.8 4.7 5.6 6.0 5.0 5.9 7.0 6.9
##     [53] 4.7 4.9 4.5 5.3 5.2 6.2 4.9 6.1 8.2 6.0 5.9 7.1 4.2
##     [66] 7.8 7.8 4.7 6.7 4.7 6.1 6.7 7.1 5.8 6.1 3.5 6.9 5.9
##     [79] 5.4 7.8 4.5 4.6 4.5 4.5 3.0 5.5 5.4 4.5 5.8 5.5 6.2
##     [92] 3.7 3.1 4.8 4.5 3.3 4.6 3.8 6.3 4.8
##
## $SERVICE
##      [1] 2.4 2.5 1.8 3.5 2.0 2.1 2.7 1.2 1.9 1.9 3.1 3.5 1.7
##     [14] 2.5 1.1 2.1 2.6 2.4 2.5 1.6 1.6 2.4 1.5 2.3 3.0 2.8
##     [27] 2.8 2.0 2.2 2.2 0.7 2.4 2.6 2.2 4.3 2.6 2.8 3.7 3.0
##     [40] 3.6 3.4 2.1 3.3 3.9 3.7 3.1 3.6 2.1 3.6 4.3 2.8 4.6
##     [53] 3.3 3.4 2.6 3.0 3.0 4.5 3.2 2.2 4.0 2.8 2.7 3.0 3.3
##     [66] 3.6 3.4 3.5 3.0 2.7 3.3 3.0 3.5 3.4 3.2 3.1 3.3 3.7
##     [79] 2.4 3.4 3.6 3.7 2.6 3.0 3.8 4.0 3.1 2.6 3.6 2.5 3.3
##     [92] 3.5 4.0 2.9 3.3 2.9 3.1 3.6 3.4 3.3
##
## $SALESFOR
##      [1] 2.3 4.0 2.3 4.5 2.8 1.4 3.7 1.7 2.5 2.5 3.0 3.4 1.1
##     [14] 2.4 2.6 2.2 2.5 1.7 2.6 2.1 3.2 2.7 3.1 3.8 2.5 2.2
##     [27] 2.3 2.5 2.5 2.1 2.1 2.7 2.9 3.0 2.7 1.9 2.2 3.2 2.5
##     [40] 2.9 1.5 1.8 2.6 3.0 2.5 2.5 2.3 2.5 2.5 3.4 3.6 4.0
##     [53] 2.2 2.6 2.3 2.5 2.3 2.9 2.7 2.6 4.4 2.8 2.7 3.8 1.4
##     [66] 4.0 4.6 3.0 2.6 1.7 3.9 2.6 3.4 2.8 3.9 1.7 3.2 2.4
##     [79] 2.6 4.6 1.3 1.4 3.2 2.4 1.4 3.0 2.5 3.1 2.3 2.7 3.9
##     [92] 1.9 1.6 1.9 2.7 1.5 2.1 2.1 4.0 2.8

```

#Si queremos conocer únicamente el nombre de las variables que contiene el #fichero de datos, podemos utilizar la función names(<nombre.del.fichero>)
names(hatco)

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

#La función str(<nombre.del.fichero>), en cambio, nos proporciona una breve descripción de las variables contenidas en el fichero

```
str(hatco)
```

```
## List of 15
## $ DELSPEED: num [1:100] 4.1 1.8 2.7 4.6 2.4 2.8 3.7 3.4 2.4 2.4 ...
## $ PRICELEV: num [1:100] 0.6 3 1 2.4 1.6 1.4 1.5 0.4 1.5 1.5 ...
## $ PRICEFLE: num [1:100] 6.9 6.3 7.1 9.5 8.8 8.1 8.6 8.3 6.7 6.6 ...
## $ MANUFIMA: num [1:100] 4.7 6.6 5.9 6.6 4.8 3.8 5.7 2.5 4.8 4.8 ...
## $ SERVICE : num [1:100] 2.4 2.5 1.8 3.5 2 2.1 2.7 1.2 1.9 1.9 ...
## $ SALESFOR: num [1:100] 2.3 4 2.3 4.5 2.8 1.4 3.7 1.7 2.5 2.5 ...
## $ PRODUCTQ: num [1:100] 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 "Small","Large": 1 2 2 1 1 2 1 1 2 2 ...
## $ USAGELEV: num [1:100] 32 43 32 46 32 39 38 35 36 36 ...
## $ SATISFLE: num [1:100] 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 "Decentralized",...: 1 2 2 1 1 2 1 1 2 2 ...
## $ INDUSTRI: Factor w/ 2 levels "Other Industries",...: 2 1 2 2 1 1 2 1 1 1 ...
## $ SITCOMP : Factor w/ 3 levels "New Task","Modified Rebuy",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ RANDZ : num [1:100] 0 0 1 0 0 0 0 1 0 1 ...
## - attr(*, "label.table")=List of 15
## ..$ DELSPEED: NULL
## ..$ PRICELEV: NULL
## ..$ PRICEFLE: NULL
## ..$ MANUFIMA: NULL
## ..$ SERVICE : NULL
## ..$ SALESFOR: NULL
## ..$ PRODUCTQ: NULL
## ..$ TAMEMP : Named num [1:2] 1 0
## .. ..- attr(*, "names")= chr [1:2] "Large" "Small"
## ..$ USAGELEV: NULL
## ..$ SATISFLE: NULL
## ..$ ESPCOMPR: Named num [1:2] 1 0
## .. ..- attr(*, "names")= chr [1:2] "Total Value Analysis" "Specification Buying"
## ..$ ESTRCOMP: Named num [1:2] 1 0
## .. ..- attr(*, "names")= chr [1:2] "Centralized" "Decentralized"
## ..$ INDUSTRI: Named num [1:2] 1 0
## .. ..- attr(*, "names")= chr [1:2] "Industry A Classification" "Other Industries"
## ..$ SITCOMP : Named num [1:3] 3 2 1
## .. ..- attr(*, "names")= chr [1:3] "Straight Rebuy" "Modified Rebuy" "New Task"
## ..$ RANDZ : NULL
## - attr(*, "variable.labels")= Named chr [1:15] "Delivery Speed" "Price Level" "Price Flexibility" "I
## ..- attr(*, "names")= chr [1:15] "DELSPEED" "PRICELEV" "PRICEFLE" "MANUFIMA" ...
```

En estas líneas de código mostramos también cómo leer los datos en el formato csv de intercambio de hojas electrónicas, donde las columnas están separadas por ; y el símbolo de los decimales es la coma, , en lugar del punto decimal anglosajón.

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
```

```
## Warning in library(package, lib.loc = lib.loc,  
## character.only = TRUE, logical.return = TRUE, : there is  
## no package called 'dplyr'
```

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

```
## [1] "DELSPEED" "PRICELEV" "PRICEFLE" "MANUFIMA" "SERVICE"  
## [6] "SALESFOR" "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)
```

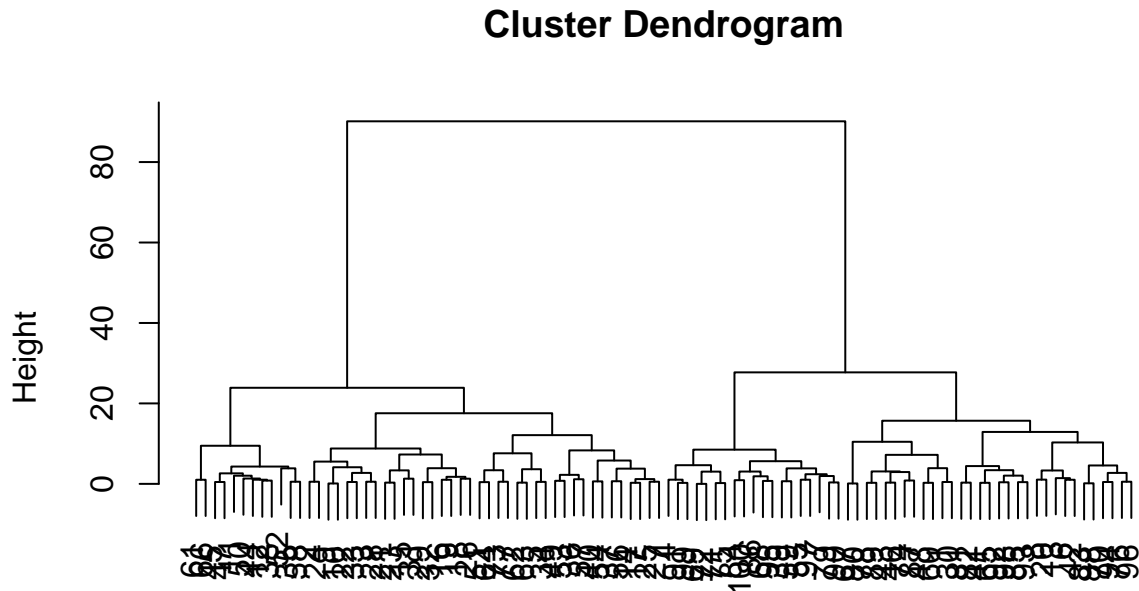
```
##           DELSPEED  PRICELEV  PRICEFLE  MANUFIMA  
## DELSPEED  1.00000000 -0.3492251  0.50929519  0.0504142  
## PRICELEV  -0.34922515  1.00000000 -0.48721259  0.2721868  
## PRICEFLE  0.50929519 -0.4872126  1.00000000 -0.1161041  
## MANUFIMA  0.05041420  0.2721868 -0.11610408  1.0000000  
## SERVICE   0.61190069  0.5129808  0.06661728  0.2986774  
## SALESFOR  0.07711522  0.1862433 -0.03431610  0.7882245  
## PRODUCTQ -0.48263094  0.4697458 -0.44811201  0.1999811  
##           SERVICE  SALESFOR  PRODUCTQ  
## DELSPEED  0.61190069  0.07711522 -0.4826309  
## PRICELEV  0.51298082  0.18624325  0.4697458  
## PRICEFLE  0.06661728 -0.03431610 -0.4481120  
## MANUFIMA  0.29867737  0.78822454  0.1999811  
## SERVICE   1.00000000  0.24080818 -0.0551613  
## SALESFOR  0.24080818  1.00000000  0.1772939  
## PRODUCTQ -0.05516130  0.17729392  1.0000000
```

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)
```



```
dist(bases, method = "euclidean")
hclust (*, "ward.D")
```

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(a
#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"
## [4] "withinss"     "tot.withinss" "betweeness"
## [7] "size"         "iter"         "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
## 1  4.382692 1.580769 8.900000 4.925000 2.957692 2.525000
## 2  2.575000 3.212500 6.804167 5.597917 2.870833 2.816667
##  PRODUCTQ
## 1  5.903846
## 2  8.127083
```

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

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

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

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 = 3.8704, df = 97.393, p-value = 0.0001966
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  3.158448 9.806295
## sample estimates:
## mean in group 1 mean in group 2
##           49.21154           42.72917
```

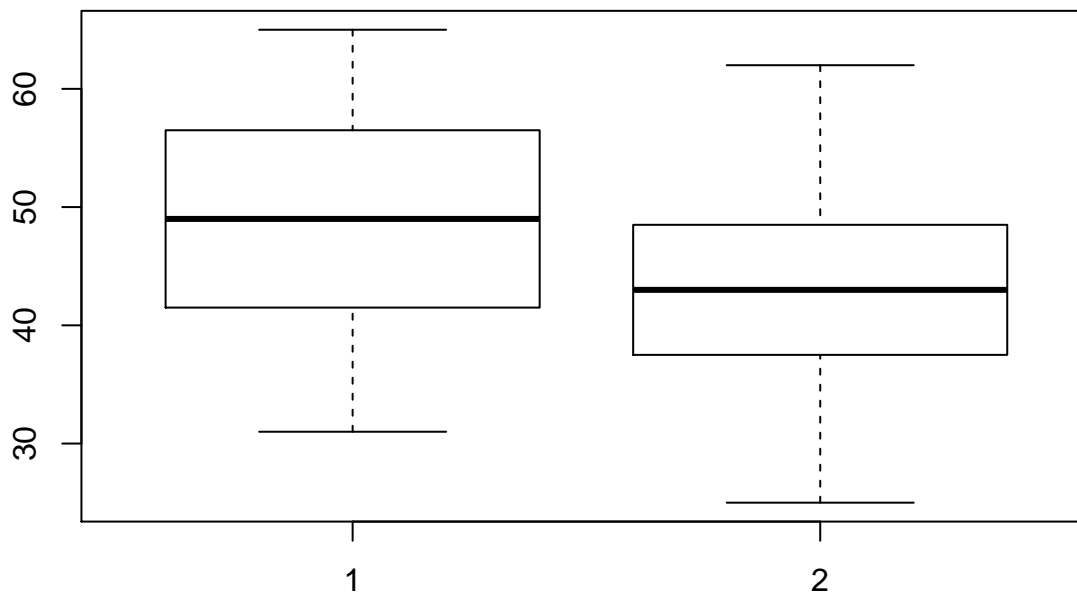
```
summary(lm(hatco$USAGELEV ~ bases.kmeans2$cluster))
```

```
##
## Call:
## lm(formula = hatco$USAGELEV ~ bases.kmeans2$cluster)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.2115  -5.9792   0.2708   6.2708  19.2708
##
## Coefficients:
##              Estimate Std. Error t value
## (Intercept)      55.694      2.633  21.152
## bases.kmeans2$cluster  -6.482      1.686  -3.846
```

```
##                                Pr(>|t|)
## (Intercept)                   < 2e-16 ***
## bases.kmeans2$cluster 0.000214 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.421 on 98 degrees of freedom
## Multiple R-squared:  0.1311, Adjusted R-squared:  0.1223
## F-statistic: 14.79 on 1 and 98 DF,  p-value: 0.0002139
```

Y ver las medias en un gráfico como el siguiente

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



Para las variables categóricas utilizamos la función table()

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

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

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

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

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

Ahora comprobáramos 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 segmentación. Visualizamos las correlaciones entre las bases de segmentación

```
cor(bases)
```

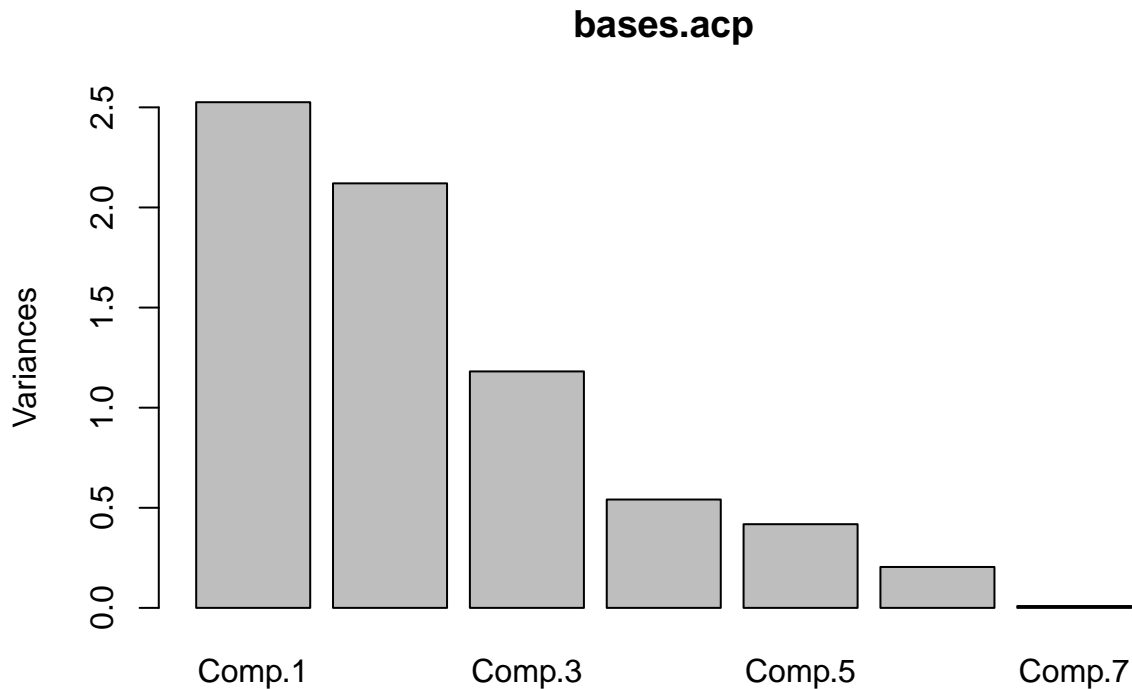
```
##          DELSPEED  PRICELEV  PRICEFLE  MANUFIMA
## DELSPEED  1.00000000 -0.3492251  0.50929519  0.0504142
## PRICELEV  -0.34922515  1.00000000 -0.48721259  0.2721868
## PRICEFLE  0.50929519 -0.4872126  1.00000000 -0.1161041
## MANUFIMA  0.05041420  0.2721868  -0.11610408  1.0000000
## SERVICE   0.61190069  0.5129808  0.06661728  0.2986774
## SALESFOR   0.07711522  0.1862433 -0.03431610  0.7882245
## PRODUCTQ  -0.48263094  0.4697458 -0.44811201  0.1999811
##          SERVICE   SALESFOR  PRODUCTQ
## DELSPEED  0.61190069  0.07711522 -0.4826309
## PRICELEV  0.51298082  0.18624325  0.4697458
## PRICEFLE  0.06661728 -0.03431610 -0.4481120
## MANUFIMA  0.29867737  0.78822454  0.1999811
## SERVICE   1.00000000  0.24080818 -0.0551613
## SALESFOR  0.24080818  1.00000000  0.1772939
## PRODUCTQ -0.05516130  0.17729392  1.0000000
```

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 resultado 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 variación 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)
```

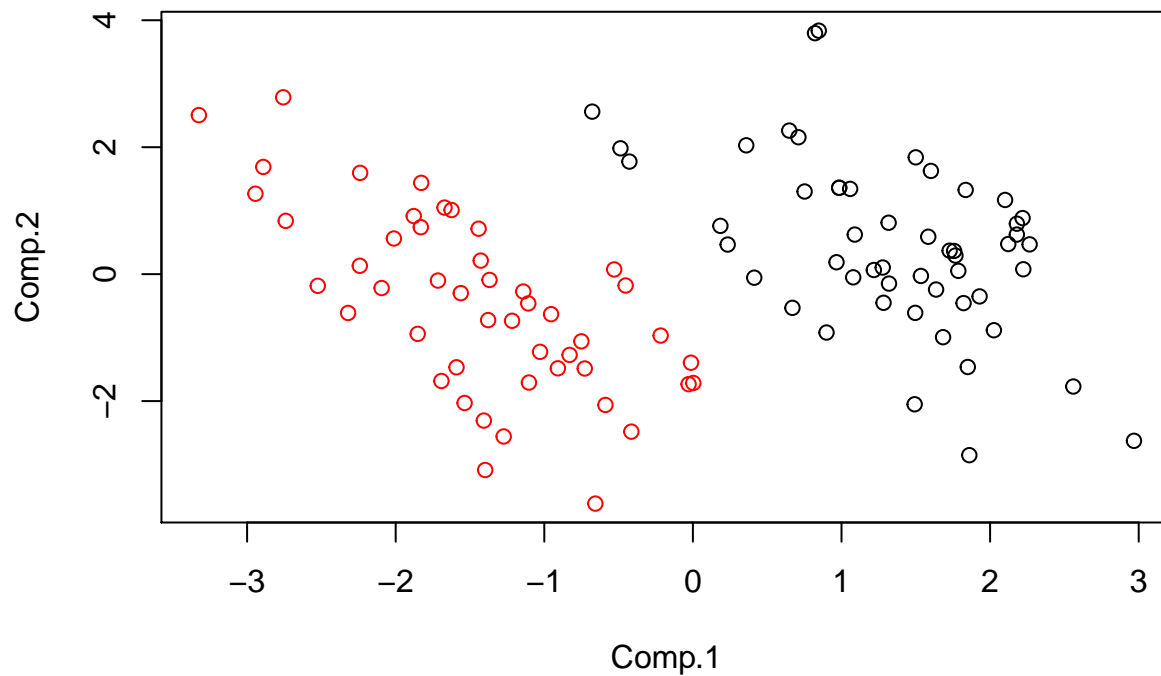
```
## Importance of components:
##               Comp.1   Comp.2   Comp.3
## Standard deviation  1.5892681 1.4561544 1.0867794
## Proportion of Variance 0.3608247 0.3029122 0.1687271
## Cumulative Proportion 0.3608247 0.6637370 0.8324640
##               Comp.4   Comp.5   Comp.6
## Standard deviation  0.73564181 0.64655249 0.45209725
## Proportion of Variance 0.07730984 0.05971859 0.02919885
## Cumulative Proportion 0.90977386 0.96949244 0.99869129
##               Comp.7
## Standard deviation  0.095712906
## Proportion of Variance 0.001308709
## Cumulative Proportion 1.000000000
```

Vemos que con 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 clasificación que hemos realizado antes, podemos utilizar los componentes de esta forma:

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



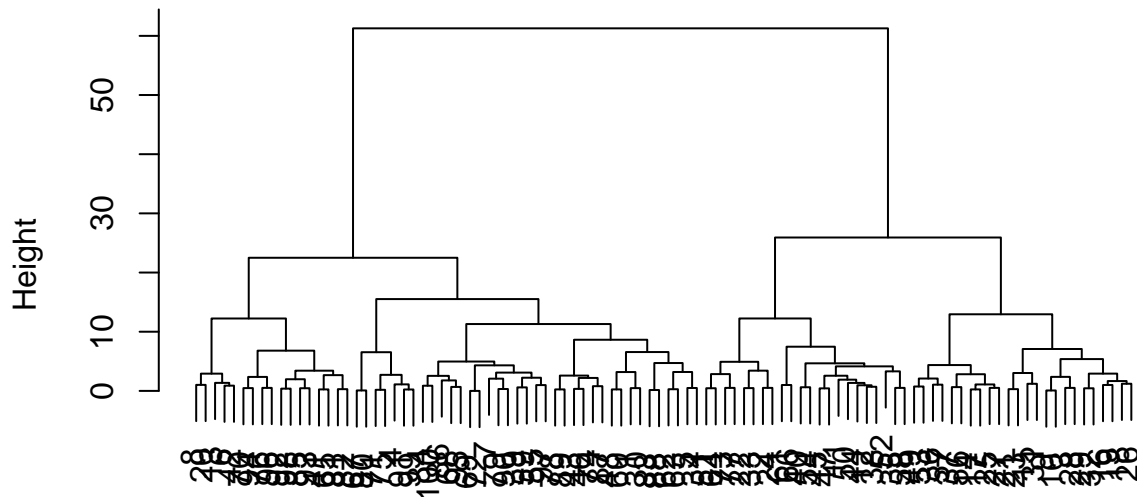
Ahora volvemos a realizar el proceso de segmentación, pero con los componentes principales. Primero la agrupación jerárquica

```
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)
```

Cluster Dendrogram



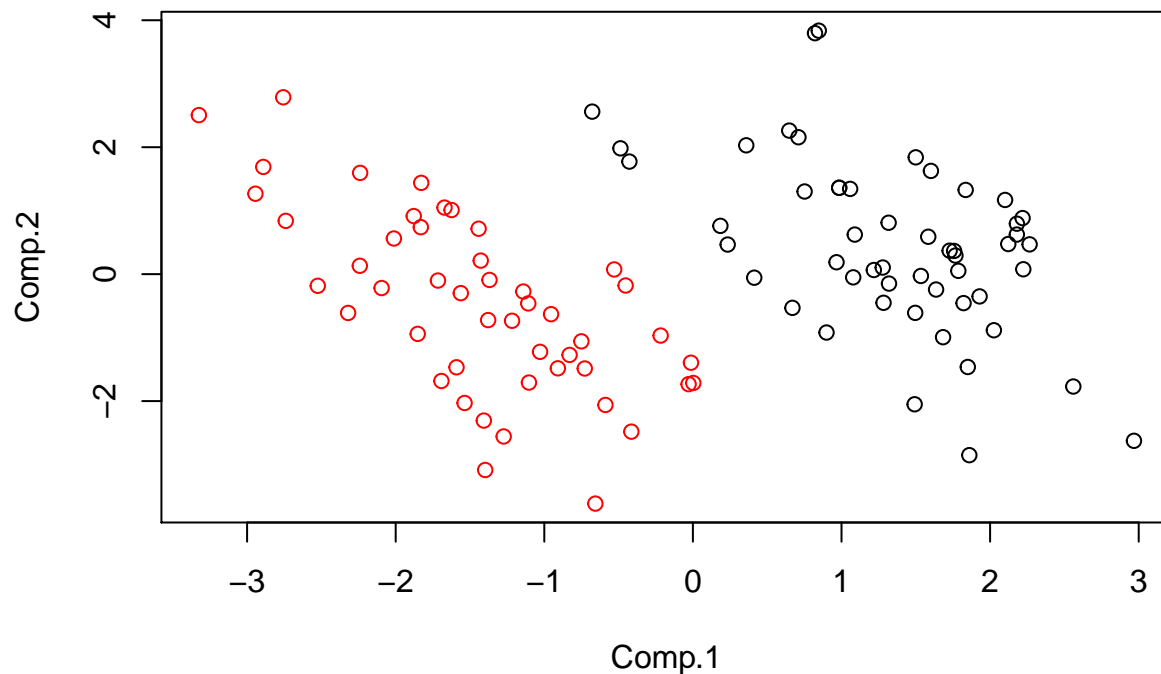
```
dist(bases.puntos)
hclust (*, "ward.D")
```

```
#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
## 1  1.263868  0.4285982  0.06955213  0.03730867
## 2 -1.483672 -0.5031370 -0.08164815 -0.04379713
##          5          6          7
## 1 -0.04070992  0.01004940  0.002705582
## 2  0.04778990 -0.01179712 -0.003176118
```

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
## 1  1.331371  0.4471037  0.05633091  0.03763319
## 2 -1.442319 -0.4843624 -0.06102516 -0.04076929
##      Comp.5      Comp.6      Comp.7
## 1 -0.005476417  0.02484801  0.002257091
## 2  0.005932785 -0.02691868 -0.002445181
##
## Clustering vector:
##  [1] 1 2 2 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2
## [27] 2 2 2 1 2 2 2 1 2 2 2 2 1 2 1 1 1 2 2 1 2 1 2 2 2
## [53] 2 2 1 2 2 2 1 1 2 2 2 2 1 2 1 1 1 1 1 2 1 1 1 2 1
## [79] 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] 251.2546 234.4954
## (between_SS / total_SS = 30.6 %)
##
## Available components:
```

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

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
```