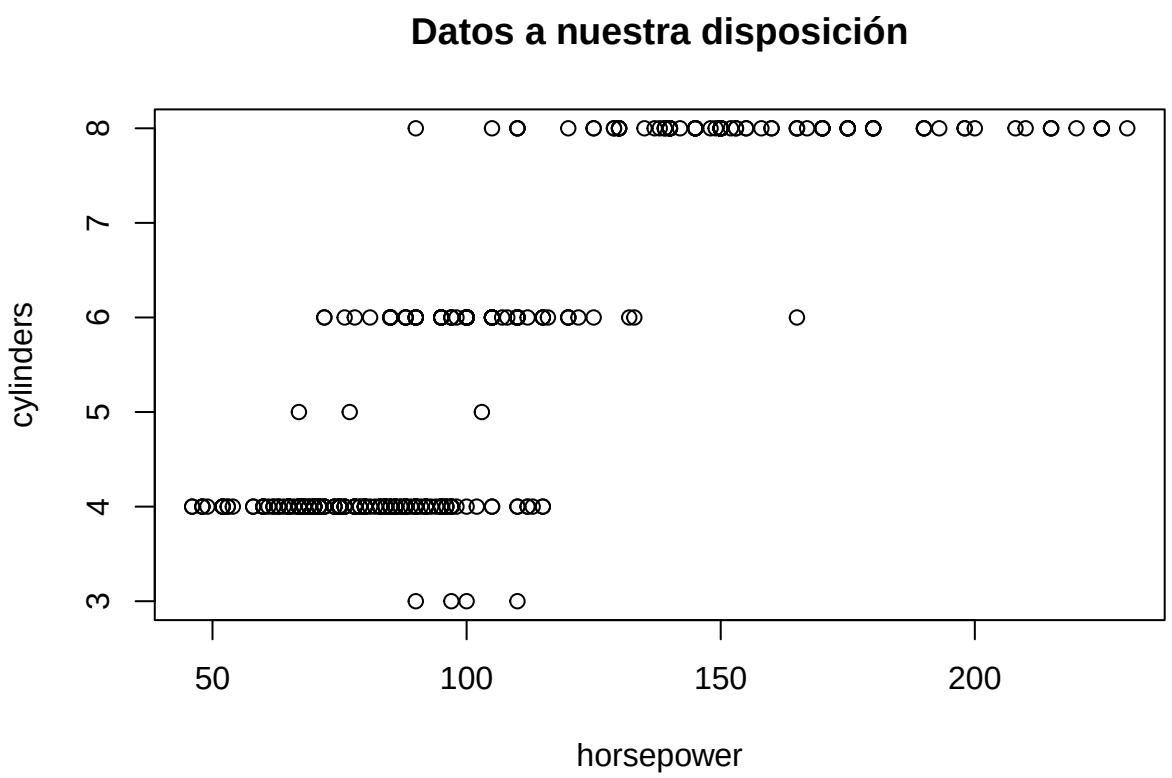
Examen de modelización

Cedric Prieels

17 de enero 2017

Preparamos primero todo el código para la resolución del ejercicio, reinicializando las variables por si hace falta y cargando la serie de datos *Auto*.

```
rm(list=ls())
par(mfrow = c(1,1))

library(ISLR)
attach(Auto)

#Pintamos los 392 datos para tener una idea de la distribución que tenemos
plot(horsepower, cylinders, main="Datos a nuestra disposición")
```




```
data <- data.frame(horsepower, cylinders)

#Los datos tienen unidades distintos, hay que escalar
data <- scale(data)
```

Parte 1

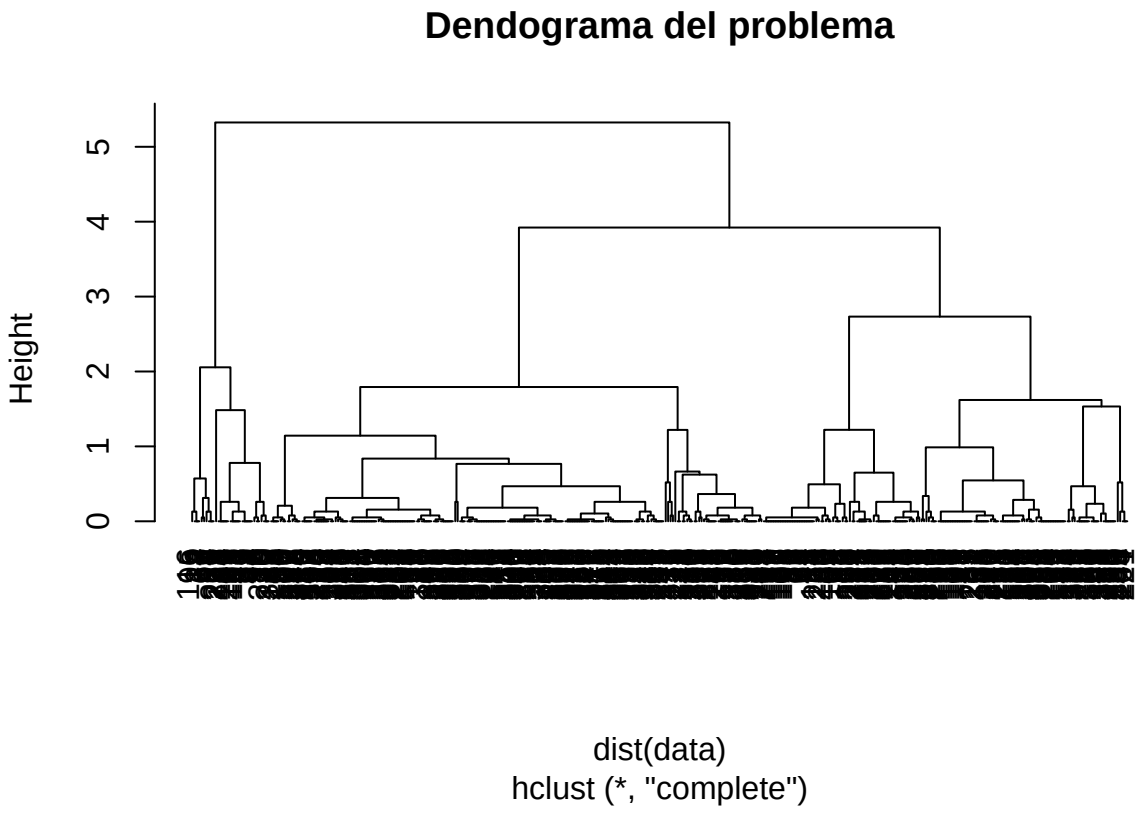
Queremos realizar un agrupamiento de los datos con el criterio de de similitud equivalente a la distancia euclidea que acabamos de definir. Vamos a usar en este caso un método de agrupamiento de tipo "Clustering

Jerárquico" que consiste en ir agrupando sucesivamente los subgrupos más similares del grupo total. La ventaja principal del agrupamienro de tipo jerárquico es que no necesitamos elegir a mano el número de clusters que queremos.

```
# Vamos a usar el método por defector de hcclust, a saber el "complete linkage"
# La función dist de R usa por defector una distancia de tipo euclidea como queremos en este caso
hc <- hclust(dist(data), method = "complete")
summary(hc)
```

```
##             Length Class  Mode
## merge       782    -none- numeric
## height      391    -none- numeric
## order       392    -none- numeric
## labels        0    -none- NULL
## method        1    -none- character
## call          3    -none- call
## dist.method   1    -none- character
```

```
plot(hc, main = "Dendograma del problema", hang = -1)
```

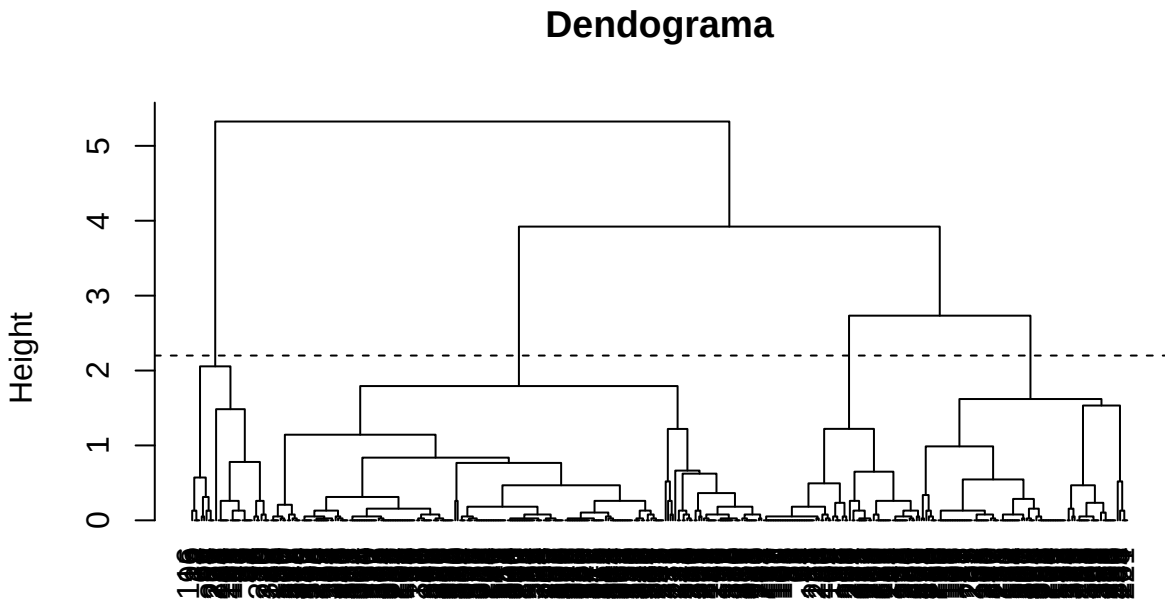


Ahora que hemos creado el dendrograma de este problema, podemos pasar a analizar los diferentes grupos que se han generado. El número de grupos que obtenemos se define a partir de una linea horizontal a una altura determinada : cuando bajamos la altura de esta linea, el número de grupos va aumentando hasta llegar a tener el mismo valor que el número de observaciones que tenemos. Como vimos en clase, "en la práctica, normalmente el número de grupos se suele seleccionar por inspección visual del dendrograma, buscándose un número de grupos razonable en función de las diferentes alturas y el número de clusters que se quiera obtener." La función *cuttree* nos ayuda en este caso. Mirando a la distribución vertical del dendograma, he decidido poner el "corte" a una altura equivalente a 70.

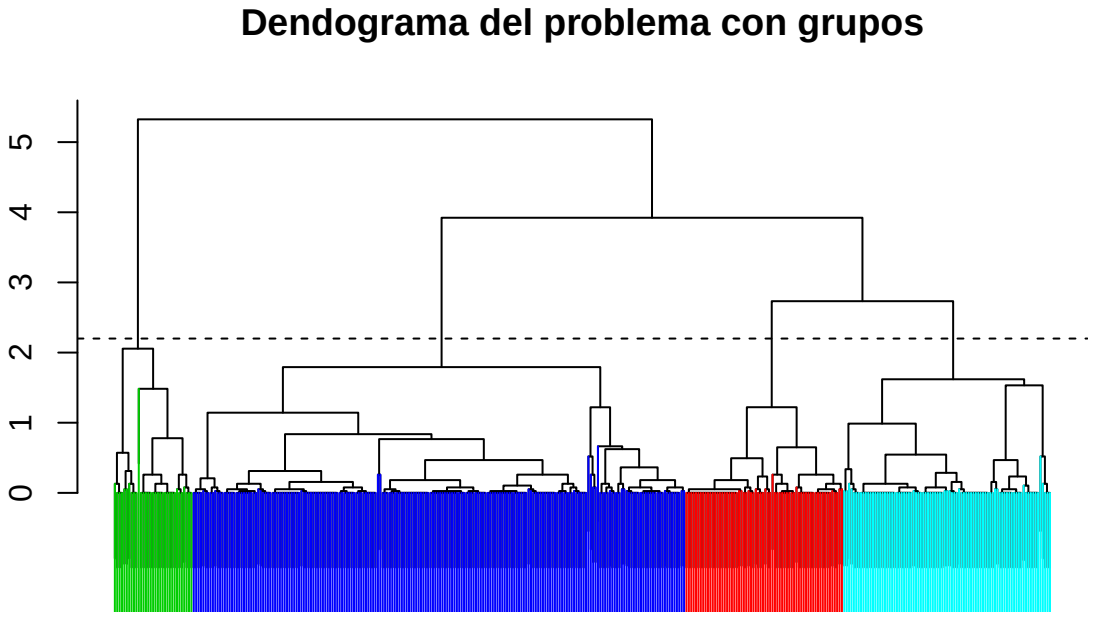
```r
#Package necesario para pintar los grupos fácilmente con la función ColorDendogram
require(sparcl)
```

```
## Loading required package: sparcl
```

```r
plot(hc, main = "Dendograma", hang = -1)
abline(h = 2.2, lty = 2)
```



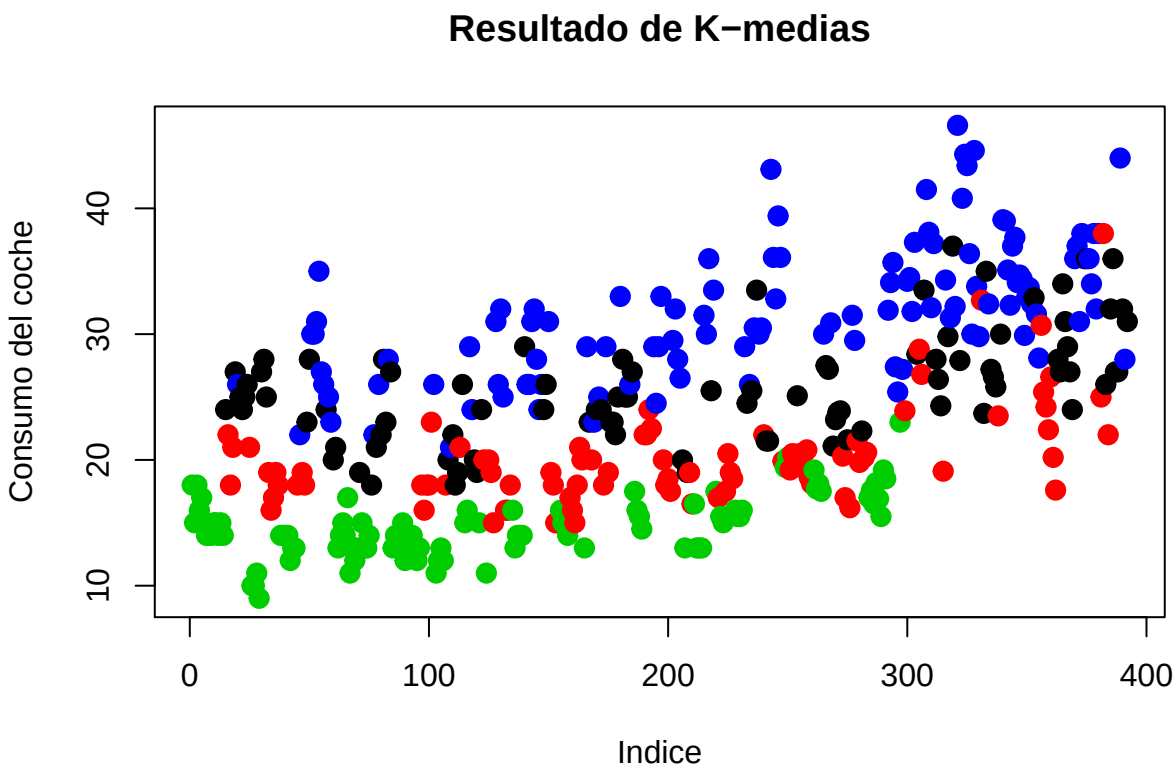

```r
height <- cutree(hc, h = 2.2)
ColorDendrogram(hc, main = "Dendograma del problema con grupos", y = height, branchlength = 10)
abline(h = 2.2, lty = 2)
```



Vemos con el código de colores que con esta altura elegida, tenemos 4 grupos distintos, que parecen bastante bien separados verticalmente (es lo que estamos buscando, porque cuanto antes se unen las ramas verticalmente, más similitud hay entre los grupos de observaciones). Vamos a intentar ahora caracterizar los diferentes grupos en función del consumo del coche, usando esta vez un método k-media, ya que tenemos una idea del número de grupos que caracterizan este problema.

Parte 2

```
km.out <- kmeans(data, centers = 4)
plot(mpg, col = (km.out$cluster), main = "Resultado de K-medias", xlab="Indice",
     ylab = "Consumo del coche",pch = 20, cex = 2)
```



Vemos que parece que los grupos se separan bastante bien con esta variable y los grupos que tenemos ahora. Podemos también determinar cuantas observaciones hay en cada cluster, el valor medio del consumo y la dispersión del consumo en cada grupo. Para esto, creamos 4 vectores que acumulan los valores de consumo del grupo correspondiente.

```
consumoGrupo1 <- c()
consumoGrupo2 <- c()
consumoGrupo3 <- c()
consumoGrupo4 <- c()

for(i in 1:length(km.out$cluster)){
  if(km.out$cluster[i] == 1) {
    consumoGrupo1 <- append(consumoGrupo1, mpg[i])
  } else if(km.out$cluster[i] == 2) {
    consumoGrupo2 <- append(consumoGrupo2, mpg[i])
  } else if(km.out$cluster[i] == 3) {
    consumoGrupo3 <- append(consumoGrupo3, mpg[i])
  } else if(km.out$cluster[i] == 4) {
    consumoGrupo4 <- append(consumoGrupo4, mpg[i])
  }
}

#Determinación del número de datos en cada grupo
size1 <- length(consumoGrupo1)
size2 <- length(consumoGrupo2)
size3 <- length(consumoGrupo3)
size4 <- length(consumoGrupo4)

#Determinación del valor medio del consumo en cada grupo
media1 <- sum(consumoGrupo1)/size1
```

```
media2 <- sum(consumoGrupo2)/size2
media3 <- sum(consumoGrupo3)/size3
media4 <- sum(consumoGrupo4)/size4

#Determinación de la dispersión del consumo en cada grupo
dispersion1 <- sd(consumoGrupo1)
dispersion2 <- sd(consumoGrupo2)
dispersion3 <- sd(consumoGrupo3)
dispersion4 <- sd(consumoGrupo4)
```

Representamos ahora los últimos resultados en una tabla, para poder compararlos.

```
resultados <- matrix(c(size1, media1, dispersion1, size2, media2, dispersion2,
                       size3, media3, dispersion3, size4, media4, dispersion4),ncol=4)

colnames(resultados) <- c("Grupo 1", "Grupo 2", "Grupo 3", "Grupo 4")
rownames(resultados) <- c("Tamaño", "Consumo medio", "Dispersión del consumo")

rtab <- as.table(resultados)
head(rtab)
```

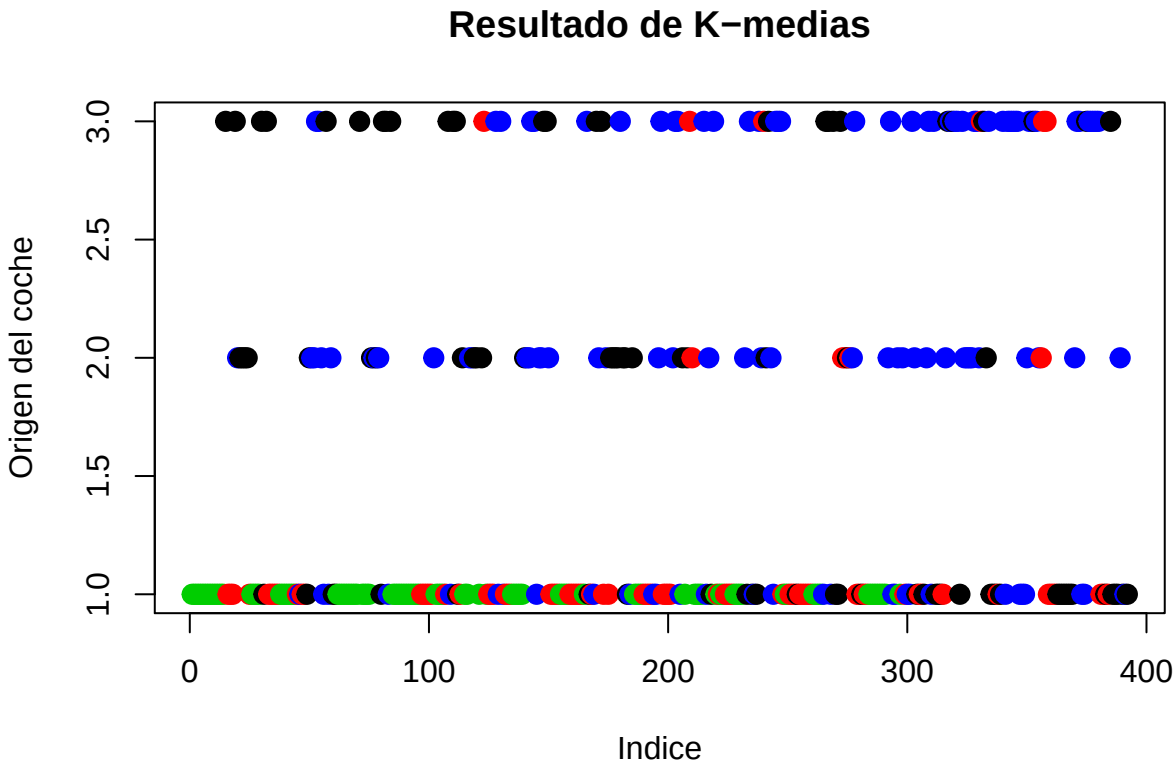
```
##                          Grupo 1    Grupo 2    Grupo 3    Grupo 4
## Tamaño                 90.000000  88.000000  99.000000 115.000000
## Consumo medio          25.680000  20.088636  14.661616  31.828696
## Dispersión del consumo  4.252936   3.811731   2.368704   5.326066
```

Como pequeña comprobación, podemos comprobar que la suma de los elements de la primera fila nos devuelve 392, el número de elementos que teníamos el principio. Vemos que el grupo que tiene el mayor número de elementos es el grupo 3, mientras que el grupo 4 tiene el menor número de datos. El valor medio de consumo más alto corresponde al grupo 2, y la mayor dispersión se encuentra en el grupo 2 también. Todos estos resultados parecen estar de acuerdo con lo que observemos en el plot anterior.

Parte 3

Podemos pensar un poco ahora para ver si existe o no alguna relación entre los grupos generados y el país de fabricación de los vehículos también.

```
plot(origin, col = (km.out$cluster), main = "Resultado de K-medias", xlab ="Indice",
     ylab = "Origen del coche", pch = 20, cex = 2)
```



En este caso vemos que no aprece que exista alguna relación entre esta variable y los grupos que hemos creado, al contrario de cuando estudiamos la variable correspondiente al consumo del coche. Todos los grupos parecen mezclados. Lo único que podemos concluir es que hay un grupo que aparece solamente cuando el origen vale 1.0 y que por lo tanto el coche viene de America (el grupo rojo, en mi caso).

Parte 4

Vamos a usar en este caso una red bayesiana de tipo datadriven (usando los métodos de tipo Hill-climbing y Tabu Search). Cargamos primero los paquetes necesarios y creamos un data frame con las variables que vamos a necesitar.

```
if (!require(bnlearn)) install.packages("bnlearn")
```

```
## Loading required package: bnlearn
```

```
##
## Attaching package: 'bnlearn'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
if (!require(RBGL)) {
      source("http://bioconductor.org/biocLite.R")
      biocLite() # Activa el conjunto de librerías que permiten realizar la instalacion remota.
      biocLite("RBGL")
}
```

```
## Loading required package: RBGL

## Loading required package: graph

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following object is masked from 'package:bnlearn':
##
##     score

## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colnames,
##     do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, lengths, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff,
##     sort, table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'graph'

## The following objects are masked from 'package:bnlearn':
##
##     degree, nodes, nodes<-
```

```
# Paquete gRain
library(gRain)
```

```
## Loading required package: gRbase

##
## Attaching package: 'gRbase'

## The following objects are masked from 'package:bnlearn':
##
##     children, parents
```

```r
library(bnlearn)

#Paquete que permite pintar las redes de una manera mejor
library("Rgraphviz")
```

```
## Loading required package: grid
```

```r
data2 <- data.frame(origin, horsepower, mpg, displacement, weight, acceleration)
#Escalamos los datos, como no tienen las mismas unidades
data2 <- data.frame(scale(data2))
```

Creamos ahora las dos redes bayesianas, las pintamos, miramos la fuerza de los enlaces en cada caso y calculamos el score de cada red, para ver cuál es la mejor.

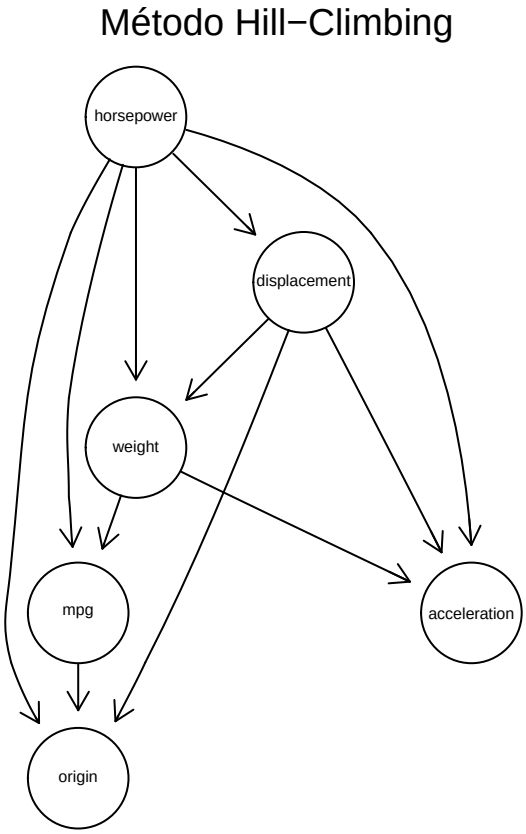
```r
par(mfrow = c(1,2))

#Hill-Climbing
hillClimbing <- hc(data2, debug = F)
graphviz.plot(hillClimbing, main="Método Hill-Climbing")
arc.strength(hillClimbing, data = data2)
```

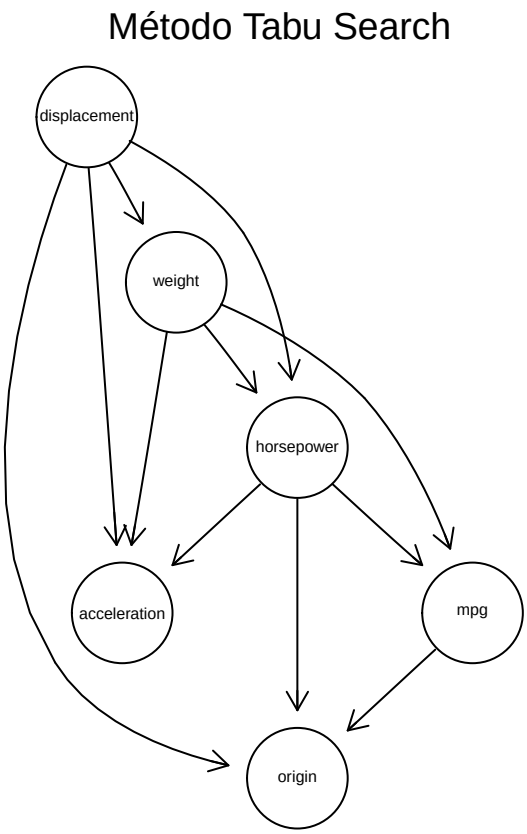
```
##            from           to    strength
## 1  displacement       weight -133.832406
## 2    horsepower displacement -317.496990
## 3        weight          mpg  -54.673593
## 4    horsepower acceleration  -97.377661
## 5  displacement       origin  -39.950816
## 6        weight acceleration  -47.294922
## 7    horsepower       origin  -18.107574
## 8           mpg       origin   -7.038461
## 9    horsepower          mpg   -5.981039
## 10 displacement acceleration   -4.041780
## 11   horsepower       weight   -2.931882
```

```r
hcScore <- bnlearn::score(hillClimbing, data = data2)
hcScore
```

```
## [1] -2111.839
```

```r
#Tabu Search
tabuSearch <- tabu(data2, debug = F)
graphviz.plot(tabuSearch, main="Método Tabu Search")
```






```
arc.strength(tabuSearch, data = data2)
```

```
##           from           to    strength
## 1  displacement       weight -397.620501
## 2        weight          mpg  -54.673593
## 3    horsepower acceleration  -97.377661
## 4  displacement       origin  -39.950816
## 5        weight acceleration  -47.294922
## 6    horsepower       origin  -18.107574
## 7           mpg       origin   -7.038461
## 8    horsepower          mpg   -5.981039
## 9  displacement acceleration   -4.041780
## 10 displacement   horsepower  -53.708895
## 11       weight   horsepower   -2.931882
```

```
tabuScore <- bnlearn::score(tabuSearch, data = data2)
tabuScore
```

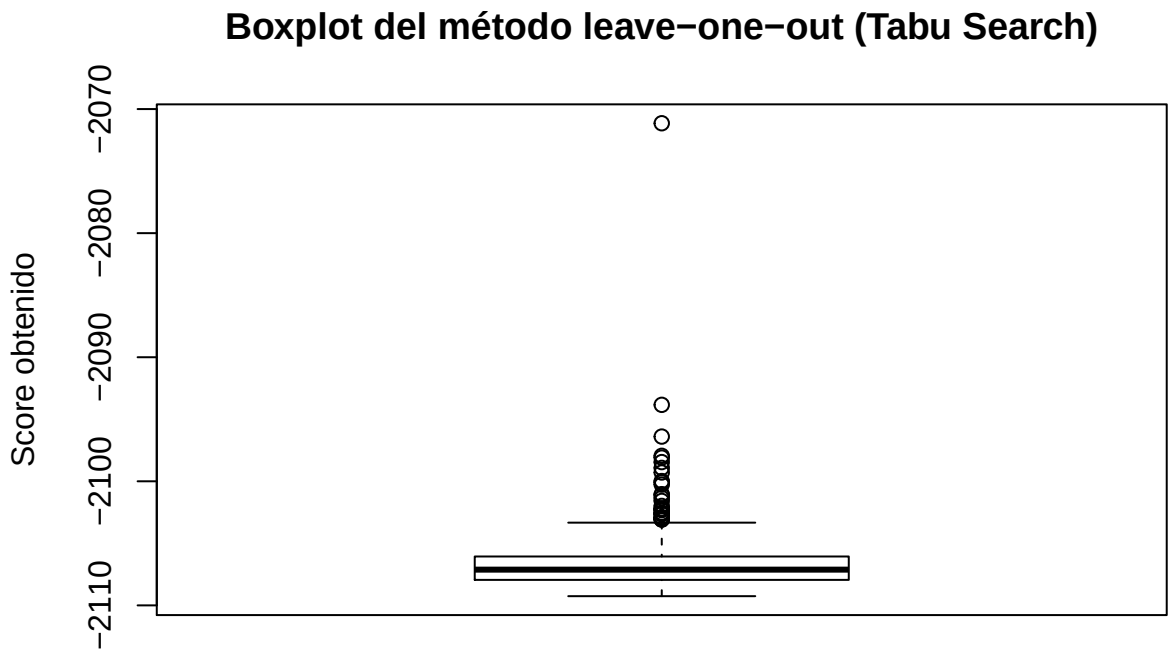
```
## [1] -2111.839
```

Vemos que en este caso los dos métodos devuelven exactamente el mismo score. Ahora podemos estimar el país de fabricación del vehiculo (por ejemplo), a partir de las otras variables, gracias a la red bayesiana, usando los métodos *setEvidence* en diferentes nodos y después *querygrain* como lo hicimos en clase.

Parte 5

Para estimar la bondad del modelo ajustado, vamos a user el método leave-one-out porque no tenemos muchos datos en este caso. Vamos a usar el método Tabu Search en esta parte del ejercicio, como tiene el mismo score que el método Hill-Climbing. El objetivo consiste en calcular el score obtenido usando todos los datos, menos uno cada vez, y ir cambiando el dato que dejamos sin usar.

```
par(mfrow = c(1,1))
n <- nrow(data2)
train <- 1:n
tabuScore <- rep(NA, n)

for (i in train){

  #Ejecutamos hc para todos los datos excepto el elemento -i
  tabuSearch <- tabu(data2[train[-i],])
  tabuScore[i] <- bnlearn::score(tabuSearch, data = data2[train[-i],])

}
boxplot(tabuScore, ylab = "Score obtenido", main="Boxplot del método leave-one-out (Tabu Search)")
```




```
summary(tabuScore, digits=7)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2109.258 -2107.946 -2107.121 -2106.519 -2106.066 -2071.144
```

Este método summary nos devuelve muchas informaciones importantes sobre el valor medio, los cuartiles y la dispersión de los scores obtenidos con este método de Leave-one-out. Vemos por ejemplo que el valor medio del score obtenido con este método vale -2106.519 en este caso.