Ayudantia 6: Clusters Jerárquicos

Actividad de Ayudantia 6

Para la actividad de esta semana deberan generar un sample (subconjunto al azar de los datos de spotify) de la data del proyecto 2 (de unas 10.000 observaciones para que sea parecido a lo trabajado aqui) (consideren que esta tarea les puede ser de gran ayuda para cuando tengan que hacer el proyecto 2) y realizar el analisis de clustering jerarquico (exploren los distintos tipos de distancia, metodos que se pueden utilizar para hacer clustering jerarquico, probar cortar el arbol para distintos h, distintos k, y va variando la cantidad de cluster segun el h que eligan, caracterizar los clusters que encuentren).

Cargar Datos Actividad

- Dado el formato con el que estan subidos los datos, para cargar los datos debemos ir a Session, luego a Load Workspace y seleccionamos el archivo .RData que subio el profe

Sample Datos Actividad

```
# Este es un ejemplo de como hacer un sample con la funcion slice_sample de dplyr, que hace muy sencillo el tomar una

# sample_spotify <- beats %>% slice_sample(n=10000)
```

Ayudantia 6

Importar Librerias

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.4

## Warning: package 'tidyr' was built under R version 4.0.4

## Warning: package 'dplyr' was built under R version 4.0.4

## -- Conflicts ------------------------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(cluster)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.5

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.0.5

##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Cargar Datos:

```
setwd("C:/Users/sebah/Desktop/U/mineria de datos/ayudantia 6")

data <- read.csv("Spotify_Songs.csv")

summary(data)
```

```
##    track_id          track_name        track_artist       track_popularity
##  Length:15630       Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  track_album_id     track_album_name   track_album_release_date
##  Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##  playlist_name      playlist_id        playlist_genre     playlist_subgenre
##  Length:15630       Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  danceability          energy               key              loudness
##  Length:15630       Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##      mode           speechiness        acousticness       instrumentalness
##  Length:15630       Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##    liveness           valence            tempo           duration_ms
##  Length:15630       Length:15630       Length:15630       Length:15630
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
head(data)
```

Pre Procesamiento de los Datos

Limpieza Datos:

Para este dataset el proceso de limpieza de datos sera un poco mas extensa por lo que debemos ir por partes

- Primero verificar la existencia de valores NA o faltantes

```
# Para las observaciones que tengan datos faltantes, le asignamos el valor NA para eliminarlos en el siguiente paso
data[data == ""] <- NA

# Verificamos donde hay valores NAs
data %>%
  summarise_all(funs(sum(is.na(.))))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

##   track_id track_name track_artist track_popularity track_album_id
## 1        0          7            8                4              5
##   track_album_name track_album_release_date playlist_name playlist_id
## 1                9                       5             6           7
##   playlist_genre playlist_subgenre danceability energy key loudness mode
## 1              8                 8            8      9  10       11   12
##   speechiness acousticness instrumentalness liveness valence tempo duration_ms
## 1          12           12               12       13      13    15          15
```

```
# De existir eliminamos todas las observaciones que presenten estos datos
data_pre <- data %>%
  filter(!(is.na(track_name)|is.na(track_artist)|is.na(track_album_name)|is.na(duration_ms)))

# Corroboramos que no queden datos NA
data_pre %>%
  summarise_all(funs(sum(is.na(.))))
```

```
##   track_id track_name track_artist track_popularity track_album_id
## 1        0          0            0                0              0
##   track_album_name track_album_release_date playlist_name playlist_id
## 1                0                       0             0           0
##   playlist_genre playlist_subgenre danceability energy key loudness mode
## 1              0                 0            0      0   0        0    0
##   speechiness acousticness instrumentalness liveness valence tempo duration_ms
## 1           0            0                0        0       0     0           0
```

- Segundo filtrar y remover datos duplicados

```
data_pre <- data_pre[!duplicated(data_pre$track_id),]
```

- Tercero verificar la existencia de errores en los datos de las observaciones

```
# Al explorar la base de datos podemos darnos cuenta de que hay varias observaciones que tiene mal ingresado los dato
# Por lo que tomaremos la columna track_popularity (como sabemos que es un valor numerico entre 0-100) y transformare
# de factor a numerico, por lo que todas las observaciones que no sean numeros se ingresara NA por defecto

data_pre$track_popularity <- as.numeric(as.character(data_pre$track_popularity))
```

```
## Warning: NAs introducidos por coerción
```

```
# Como generamos nuevos valores NA dentro de nuestra BBDD, debemos volver a ejecutar el paso uno de la limpieza de da

data_pre <- data_pre %>%
  filter(!(is.na(track_popularity)))

# Eliminamos el patron <U que aparece en algunas observaciones en track_name y track_artist

data_pre <- data_pre[!grepl("<U",data_pre$track_name),]
data_pre <- data_pre[!grepl("<U",data_pre$track_artist),]

# Ahora corroboraremos si existen canciones que esten duplicadas
data_pre %>% count(duplicated(data_pre$track_name))
```

```
##   duplicated(data_pre$track_name)     n
## 1                           FALSE 12244
## 2                            TRUE  1623
```

```
# Como existen canciones repetidas realizamos la consulta para obtener los valores distintos, pero este hecho obvia q
data_pre %>% distinct(track_name, .keep_all = TRUE, ) %>% head()
```

```
##                track_id                                          track_name
## 1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury Remix
## 2 0r7CVbZTWZgbTCYdfa2P31                       Memories - Dillon Francis Remix
## 3 1z1Hg7Vb0AhHDiEmnDE79l                      All the Time - Don Diablo Remix
## 4 75FpbthrwQmzHlBJLuGdC7                    Call You Mine - Keanu Silva Remix
## 5 1e8PAfcKUYoKkxPhrHqw4x                Someone You Loved - Future Humans Remix
## 6 7fvUMiyapMsRRxr07cU8Ef     Beautiful People (feat. Khalid) - Jack Wins Remix
##       track_artist track_popularity         track_album_id
## 1      Ed Sheeran               66 2oCs0DGTsRO98Gh5ZSl2Cx
## 2        Maroon 5               67 63rPSO264uRjW1X5E6cWv6
## 3    Zara Larsson               70 1HoSmj2eLcsrR0vE9gThr4
## 4 The Chainsmokers              60 1nqYsOef1yKKuGOVchbsk6
## 5   Lewis Capaldi               69 7m7vv9wlQ4i0LFuJiE2zsQ
## 6      Ed Sheeran               67 2yiy9cd2QktrNvWC2EUi0k
##                                  track_album_name
## 1 I Don't Care (with Justin Bieber) [Loud Luxury Remix]
## 2                       Memories (Dillon Francis Remix)
## 3                       All the Time (Don Diablo Remix)
## 4                          Call You Mine - The Remixes
## 5                Someone You Loved (Future Humans Remix)
## 6     Beautiful People (feat. Khalid) [Jack Wins Remix]
##   track_album_release_date playlist_name           playlist_id playlist_genre
## 1               2019-06-14     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 2               2019-12-13     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 3               2019-07-05     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 4               2019-07-19     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 5               2019-03-05     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 6               2019-07-11     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
##   playlist_subgenre danceability energy key loudness mode speechiness
## 1         dance pop        0.748  0.916   6   -2.634    1      0.0583
## 2         dance pop        0.726  0.815  11   -4.969    1      0.0373
## 3         dance pop        0.675  0.931   1   -3.432    0      0.0742
## 4         dance pop        0.718   0.93   7   -3.778    1       0.102
## 5         dance pop         0.65  0.833   1   -4.672    1      0.0359
## 6         dance pop        0.675  0.919   8   -5.385    1       0.127
##   acousticness instrumentalness liveness valence   tempo duration_ms
## 1        0.102                0   0.0653   0.518 122.036      194754
## 2       0.0724          0.00421    0.357   0.693  99.972      162600
## 3       0.0794         2.33e-05     0.11   0.613 124.008      176616
```

```
## 1        0.102                0   0.0653   0.518 122.036      194754
## 2       0.0724          0.00421    0.357   0.693  99.972      162600
## 3       0.0794         2.33e-05     0.11   0.613 124.008      176616
## 4       0.0287         9.43e-06    0.204   0.277 121.956      169093
## 5       0.0803                0   0.0833   0.725 123.976      189052
## 6       0.0799                0    0.143   0.585 124.982      163049
```

```r
# Por lo que creamos una variables que almacene si existe duplicidad en la cacion y/o en el artista
data_pre$duplicate <- duplicated(data_pre[,c("track_name", "track_artist")])

# Generamos un sub data frame que almacenara solo los valores que haya obtenido el valor TRUE a la consulta anterior
data_dupli <- data_pre %>%
  filter(data_pre$duplicate == TRUE) %>%
  arrange("track_name", "track_popularity", desc(track_popularity))

# Seleciono las filas que sean distintas, borro todas las canciones que se repiten y me quedo con la mayor track popu
data_dupli <- data_dupli %>%
  distinct(track_name, track_artist, .keep_all = TRUE)

# Elimino de mi data pre procesada los datos que dieron positivo a la duplicidad, para que al momento de re insertar
data_pre <- data_pre[!(data_pre$duplicate == TRUE),]

# Junto la data pre procesada con los datos que sobrevivieron a la limpieza de duplicidad
data_pre <- rbind(data_pre, data_dupli)

# Elimino la columna que me indicaba duplicidad ya que no sera util mas adelante
data_pre$duplicate <- NULL
```

Una vez limpiados los datos, el siguiente paso en el pre procesamiento será escalar los datos pero antes debemos revisar los datos por si hay que transformar alguna variable

Revisar Estructura Datos

```r
# Transformamos cada variables al tipo de variable que sale en el archivo .txt con la descripcion de cada una

data_pre$track_id <- as.character(data_pre$track_id)
data_pre$track_name <- as.character(data_pre$track_name)
data_pre$track_artist <- as.character(data_pre$track_artist)
data_pre$track_album_id <- as.character(data_pre$track_album_id)
data_pre$track_album_name <-  as.character(data_pre$track_album_name)
data_pre$playlist_name <- as.character(data_pre$playlist_name)
data_pre$playlist_id <- as.character(data_pre$playlist_id)
data_pre$playlist_genre <- as.character(data_pre$playlist_genre)
data_pre$playlist_subgenre <- as.character(data_pre$playlist_subgenre)

data_pre$danceability <- as.double(as.character(data_pre$danceability))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$energy <- as.double(as.character(data_pre$energy))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$key <- as.double(as.character(data_pre$key))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$loudness <- as.double(as.character(data_pre$loudness))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$mode <- as.double(as.character(data_pre$mode))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$speechiness <- as.double(as.character(data_pre$speechiness))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$acousticness <- as.double(as.character(data_pre$acousticness))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$instrumentalness <- as.double(as.character(data_pre$instrumentalness))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$liveness <- as.double(as.character(data_pre$liveness))
```

```
## Warning: NAs introducidos por coerción
```

```r
data_pre$valence <- as.double(as.character(data_pre$valence))
```

```
## Warning: NAs introducidos por coerción
```

```
data_pre$tempo <- as.double(as.character(data_pre$tempo))
```

```
## Warning: NAs introducidos por coerción
```

```
data_pre$duration_ms <- as.double(as.character(data_pre$duration_ms))
```

```
## Warning: NAs introducidos por coerción
```

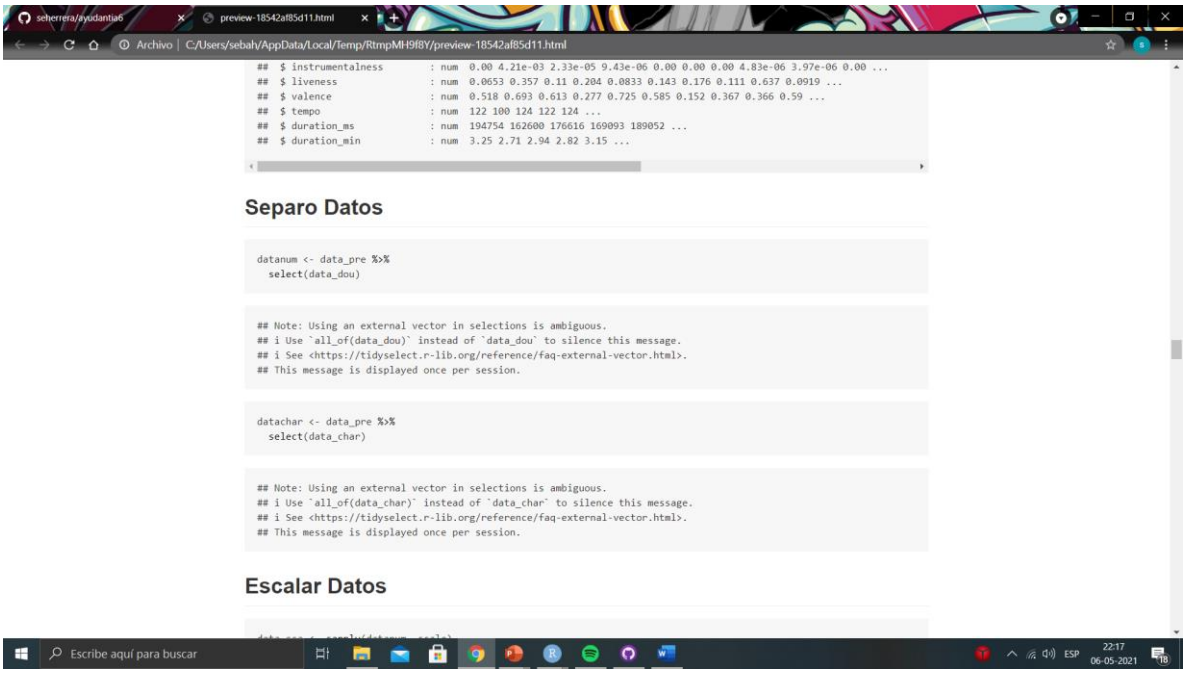
```
# transformacion de milisegundos a minutos
data_pre <- data_pre %>% mutate(duration_min = data_pre$duration_ms/60000)

# Character
data_char <- c("track_id", "track_name", "track_artist", "track_album_id", "track_album_name", "playlist_name", "play

# Double
data_dou <- c("track_popularity","danceability", "energy", "key", "loudness", "mode", "speechiness", "acousticness",

# Volvemos a borrar los datos que puedan haber quedado como NA con el cambio de tipo de variable
data_pre <- data_pre %>%
  filter(!(is.na(key)|is.na(danceability)))

summary(data_pre)
```

```
##    track_id          track_name        track_artist       track_popularity
##  Length:13735       Length:13735       Length:13735       Min.   :  0.00
##  Class :character   Class :character   Class :character   1st Qu.: 24.00
##  Mode  :character   Mode  :character   Mode  :character   Median : 45.00
##                                                           Mean   : 41.54
##                                                           3rd Qu.: 61.00
##                                                           Max.   :100.00
##  track_album_id     track_album_name   track_album_release_date
##  Length:13735       Length:13735       Length:13735
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##  playlist_name      playlist_id        playlist_genre     playlist_subgenre
##  Length:13735       Length:13735       Length:13735       Length:13735
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   danceability        energy              key            loudness
##  Min.   :0.0000   Min.   :0.000175   Min.   : 0.000   Min.   :-46.448
##  1st Qu.:0.5680   1st Qu.:0.605000   1st Qu.: 2.000   1st Qu.: -7.963
##  Median :0.6790   Median :0.733000   Median : 6.000   Median : -6.037
##  Mean   :0.6594   Mean   :0.714072   Mean   : 5.385   Mean   : -6.575
##  3rd Qu.:0.7660   3rd Qu.:0.847000   3rd Qu.: 9.000   3rd Qu.: -4.576
##  Max.   :0.9790   Max.   :1.000000   Max.   :11.000   Max.   :  1.275
##       mode         speechiness      acousticness    instrumentalness
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000000
##  1st Qu.:0.0000   1st Qu.:0.0418   1st Qu.:0.0161   1st Qu.:0.000000
##  Median :1.0000   Median :0.0646   Median :0.0829   Median :0.000008
##  Mean   :0.5641   Mean   :0.1102   Mean   :0.1679   Mean   :0.072279
##  3rd Qu.:1.0000   3rd Qu.:0.1380   3rd Qu.:0.2460   3rd Qu.:0.002440
##  Max.   :1.0000   Max.   :0.8770   Max.   :0.9940   Max.   :0.994000
##     liveness         valence           tempo         duration_ms
##  Min.   :0.0000   Min.   :0.0000   Min.   :  0.00   Min.   :  4000
##  1st Qu.:0.0930   1st Qu.:0.3390   1st Qu.: 99.94   1st Qu.:187047
##  Median :0.1200   Median :0.5300   Median :121.97   Median :214587
```

```
##  1st Qu.:0.0930   1st Qu.:0.3390   1st Qu.: 99.94   1st Qu.:187047
##  Median :0.1290   Median :0.5200   Median :121.97   Median :214587
##  Mean   :0.1939   Mean   :0.5186   Mean   :121.65   Mean   :223882
##  3rd Qu.:0.2560   3rd Qu.:0.7020   3rd Qu.:135.43   3rd Qu.:249827
##  Max.   :0.9960   Max.   :0.9790   Max.   :239.44   Max.   :517810
##   duration_min
##  Min.   :0.06667
##  1st Qu.:3.11745
##  Median :3.57645
##  Mean   :3.73137
##  3rd Qu.:4.16378
##  Max.   :8.63017
```

```
str(data_pre)
```

```
## 'data.frame':    13735 obs. of  24 variables:
##  $ track_id               : chr  "6f807x0ima9a1j3VPbc7VN" "0r7CVbZTWZgbTCYdfa2P31" "1z1Hg7Vb0AhHDiEmnDE79l" "75Fp
##  $ track_name             : chr  "I Don't Care (with Justin Bieber) - Loud Luxury Remix" "Memories - Dillon Franc
##  $ track_artist           : chr  "Ed Sheeran" "Maroon 5" "Zara Larsson" "The Chainsmokers" ...
##  $ track_popularity       : num  66 67 70 60 69 67 62 69 68 67 ...
##  $ track_album_id         : chr  "2oCs0DGTsRO98Gh5ZSl2Cx" "63rPSO264uRjW1X5E6cWv6" "1HoSmj2eLcsrR0vE9gThr4" "1nqY
##  $ track_album_name       : chr  "I Don't Care (with Justin Bieber) [Loud Luxury Remix]" "Memories (Dillon Franci
##  $ track_album_release_date: chr  "2019-06-14" "2019-12-13" "2019-07-05" "2019-07-19" ...
##  $ playlist_name          : chr  "Pop Remix" "Pop Remix" "Pop Remix" "Pop Remix" ...
##  $ playlist_id            : chr  "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9
##  $ playlist_genre         : chr  "pop" "pop" "pop" "pop" ...
##  $ playlist_subgenre      : chr  "dance pop" "dance pop" "dance pop" "dance pop" ...
##  $ danceability           : num  0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542 0.594 0.642 ...
##  $ energy                 : num  0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903 0.935 0.818 ...
##  $ key                    : num  6 11 1 7 1 8 5 4 8 2 ...
##  $ loudness               : num  -2.63 -4.97 -3.43 -3.78 -4.67 ...
##  $ mode                   : num  1 1 0 1 1 1 0 0 1 1 ...
##  $ speechiness            : num  0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623 0.0434 0.0565 0.032 ...
##  $ acousticness           : num  0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187 0.0335 0.0249 0.0567 ...
##  $ instrumentalness       : num  0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00 4.83e-06 3.97e-06 0.00 ...
##  $ liveness               : num  0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111 0.637 0.0919 ...
```

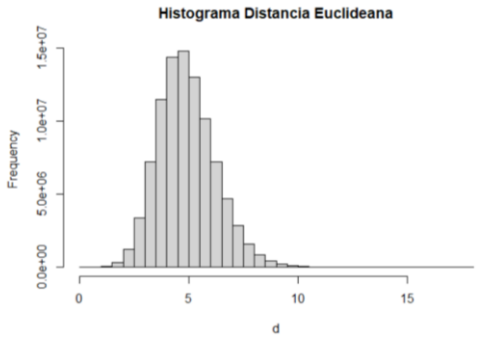


Procesamiento de los Datos

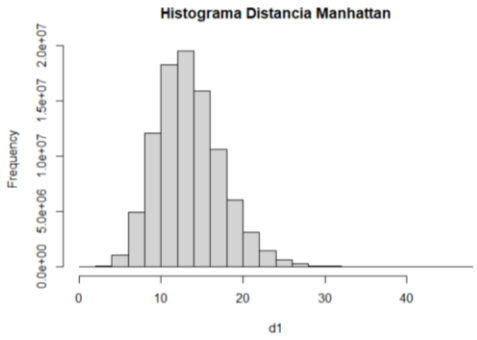
Clustering Jerarquico

- Matriz de Distancias

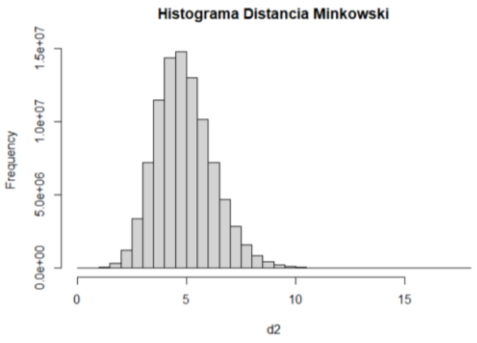
```
#Distancia Euclideana
d = dist(data_sca, method = "euclidean")

#Distancia Manhattan
d1 = dist(data_sca, method = "manhattan")

#Distancia Minkowski
d2 = dist(data_sca, method = "minkowski")

hist(d, main = "Histograma Distancia Euclideana")
```




```
hist(d1, main = "Histograma Distancia Manhattan")
```




```
hist(d2, main = "Histograma Distancia Minkowski")
```



Clustering Aglomerativo

Utilizando la funcion de R base hclust, aplicamos hierarchical clustering, a partir de la matriz de distancias d, y utilizamos el criterio complete linkage

- Complete Model

```
# Fijamos una seed para que cada vez que ejecutemos el codigo obtengamos los mismos valores y no vayan cambiado cada
set.seed(370)

model_complete <- hclust(d, method = "complete")

summary(model_complete)
```

```
##             Length Class  Mode
## merge       27468  -none- numeric
## height      13734  -none- numeric
## order       13735  -none- numeric
## labels          0  -none- NULL
## method          1  -none- character
## call            3  -none- call
## dist.method     1  -none- character
```

- Ward Model

```
set.seed(370)

model_ward <- hclust(d, method = "ward.D")

summary(model_ward)
```

```
##             Length Class  Mode
## merge       27468  -none- numeric
## height      13734  -none- numeric
## order       13735  -none- numeric
## labels          0  -none- NULL
## method          1  -none- character
## call            3  -none- call
## dist.method     1  -none- character
```

- Otra forma de hacer clustering jerarquico con la funcion agnes

```
#model_comag <- agnes(d, method = "complete")

#model_comag$ac
```

```
#model_wardag <- agnes(d, method = "ward.D")

#model_wardag$ac
```

- Comparacion de los coeficientes de aglomeracion para cada metodo

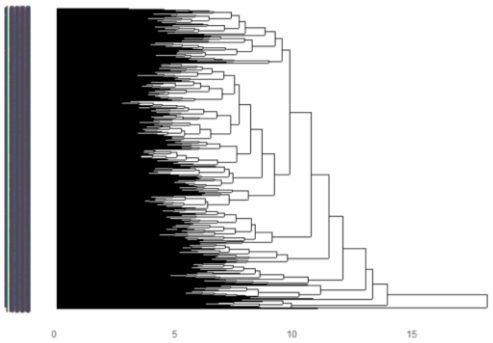
```
models <- c("single", "complete", "average", "ward")
names(models) <- c("single", "complete", "average", "ward")

#models <- c("complete", "ward")
#names(models) <- c("complete", "ward")

#agcoef <- function(x) {
 # agnes(data_sca, method = x)$ac
#}

#sapply(models, agcoef)
```

Generamos un dendrograma para visualizar la jerarquia. La libreria 'ggdendro' permite hacer estos diagramas en una sintaxis equivalente a ggplot.

```
library("ggdendro")
```

```
## Warning: package 'ggdendro' was built under R version 4.0.5
```

```
ggdendrogram(model_complete, rotate = TRUE, theme_dendro = TRUE)
```



Corte

```
# Determinamos un valor para h lo que nos entregara un valor distinto de k para cada h que escogamos, tambien podemos
groups <- cutree(model_complete, h = 10)

# Se imprimen los tamaños de cada cluster
table(groups)
```

```
## groups
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 9187  512 1585  908  385  748  210   61   55   36   23    9    1    7    8
```

```
# Generamos una nueva columna para almacenar a que cluster pertenece cada observacion (tanto en data_pre y datanum)
data_pre$clust <- as.factor(groups)
datanum$clust <- as.factor(groups)

# Graficamos las observaciones agrupadas por su cluster
fviz_cluster(list(data = data_sca, cluster = groups))
```

Caracteristicas de los clusters encontrados

```r
datanum$clust <- as.numeric(as.character(datanum$clust))

# Generamos una tabla que almacenara los valores promedios para cada uno de los clusters encontrados lo que nos permi
infoclusters <- aggregate(datanum, by=list(cluster=datanum$clust), mean)

# Borramos la columna clust ya que se repite esa informacion en la tabla generada
infoclusters$clust <- NULL

# Transformamos el tiempo de la cancion a minutos
infoclusters <- infoclusters %>% mutate(duration_min = infoclusters$duration_ms/60000)

# Borramos la columna de la duracion en milisegundoss
infoclusters$duration_ms <- NULL

infoclusters
```

```
##    cluster track_popularity danceability    energy      key   loudness
## 1        1         43.17655    0.6655525 0.7578630 5.278655  -5.621984
## 2        2         39.84766    0.6246680 0.7913848 5.210938  -5.544109
## 3        3         39.33817    0.7316858 0.5808540 6.450473  -9.060425
## 4        4         30.06167    0.6625077 0.7849725 5.205947  -7.068180
## 5        5         41.09351    0.5696636 0.3482340 4.968831 -12.575135
## 6        6         43.12567    0.5310615 0.5413984 4.823529  -8.910016
## 7        7         44.32857    0.6748524 0.6990190 6.100000  -7.039743
## 8        8         44.21311    0.6554918 0.5721639 5.885246  -7.603344
## 9        9         24.52727    0.4601455 0.4858727 4.745455 -10.956564
## 10      10         15.58333    0.3942222 0.7994444 5.583333  -7.835111
## 11      11         30.52174    0.6048261 0.6902609 3.956522  -8.110913
## 12      12         33.44444    0.6943333 0.6200000 5.111111  -9.810444
## 13      13          0.00000    0.0000000 0.3150000 1.000000 -26.087000
## 14      14         58.42857    0.1705857 0.9195714 6.285714 -17.368000
## 15      15         45.87500    0.2122125 0.2810969 5.500000 -35.390125
##         mode speechiness acousticness instrumentalness  liveness     valence
## 1  0.5641668  0.11023493   0.14330337       0.013012077 0.1800803 0.547665745
## 2  0.5546875  0.09238594   0.10696736       0.074748278 0.6342168 0.464561914
## 3  0.5009464  0.15571886   0.17013462       0.015810692 0.1430357 0.560321577
## 4  0.5881057  0.07216244   0.07087886       0.745484581 0.1656719 0.384046145
## 5  0.7012987  0.05743974   0.68510518       0.215723529 0.1476569 0.369210649
## 6  0.5882353  0.05496511   0.30675259       0.025345940 0.1463247 0.365755080
## 7  0.4952381  0.12355762   0.13008954       0.028747991 0.2134033 0.520031905
## 8  0.6065574  0.41231148   0.57569344       0.019042014 0.1687902 0.485237705
## 9  0.9090909  0.06782545   0.51323582       0.029823940 0.6774727 0.533101818
## 10 0.6388889  0.06009722   0.05184096       0.132936862 0.9188611 0.363138889
## 11 0.8260870  0.66513043   0.12960261       0.006335343 0.2562522 0.545117391
## 12 0.3333333  0.39477778   0.61422222       0.830000000 0.2234333 0.764666667
## 13 1.0000000  0.00000000   0.00000000       0.000000000 0.0000000 0.000000000
## 14 0.2857143  0.09072857   0.53915461       0.895285714 0.7680000 0.004264286
## 15 0.5000000  0.10825000   0.37528875       0.433159775 0.1848750 0.077312500
##        tempo duration_min
## 1  121.12681   3.62820745
## 2  118.81798   3.66318538
## 3  122.36159   3.43480091
## 4  126.04303   4.27521808
## 5  120.04546   3.45767290
## 6  125.37157   4.43974557
## 7  121.32324   6.29268690
## 8  108.82777   3.06215902
## 9  120.01955   3.71676939
## 10 130.67097   5.71512130
```


```
## 4        4         30.06167    0.6625077 0.7849725 5.205947  -7.068180
## 5        5         41.09351    0.5696636 0.3482340 4.968831 -12.575135
## 6        6         43.12567    0.5310615 0.5413984 4.823529  -8.910016
## 7        7         44.32857    0.6748524 0.6990190 6.100000  -7.039743
## 8        8         44.21311    0.6554918 0.5721639 5.885246  -7.603344
## 9        9         24.52727    0.4601455 0.4858727 4.745455 -10.956564
## 10      10         15.58333    0.3942222 0.7994444 5.583333  -7.835111
## 11      11         30.52174    0.6048261 0.6902609 3.956522  -8.110913
## 12      12         33.44444    0.6943333 0.6200000 5.111111  -9.810444
## 13      13          0.00000    0.0000000 0.3150000 1.000000 -26.087000
## 14      14         58.42857    0.1705857 0.9195714 6.285714 -17.368000
## 15      15         45.87500    0.2122125 0.2810969 5.500000 -35.390125
##         mode speechiness acousticness instrumentalness  liveness     valence
## 1  0.5641668  0.11023493   0.14330337       0.013012077 0.1800803 0.547665745
## 2  0.5546875  0.09238594   0.10696736       0.074748278 0.6342168 0.464561914
## 3  0.5009464  0.15571886   0.17013462       0.015810692 0.1430357 0.560321577
## 4  0.5881057  0.07216244   0.07087886       0.745484581 0.1656719 0.384046145
## 5  0.7012987  0.05743974   0.68510518       0.215723529 0.1476569 0.369210649
## 6  0.5882353  0.05496511   0.30675259       0.025345940 0.1463247 0.365755080
## 7  0.4952381  0.12355762   0.13008954       0.028747991 0.2134033 0.520031905
## 8  0.6065574  0.41231148   0.57569344       0.019042014 0.1687902 0.485237705
## 9  0.9090909  0.06782545   0.51323582       0.029823940 0.6774727 0.533101818
## 10 0.6388889  0.06009722   0.05184096       0.132936862 0.9188611 0.363138889
## 11 0.8260870  0.66513043   0.12960261       0.006335343 0.2562522 0.545117391
## 12 0.3333333  0.39477778   0.61422222       0.830000000 0.2234333 0.764666667
## 13 1.0000000  0.00000000   0.00000000       0.000000000 0.0000000 0.000000000
## 14 0.2857143  0.09072857   0.53915461       0.895285714 0.7680000 0.004264286
## 15 0.5000000  0.10825000   0.37528875       0.433159775 0.1848750 0.077312500
##        tempo duration_min
## 1  121.12681   3.62820745
## 2  118.81798   3.66318538
## 3  122.36159   3.43480091
## 4  126.04303   4.27521808
## 5  120.04546   3.45767290
## 6  125.37157   4.43974557
## 7  121.32324   6.29268690
## 8  108.82777   3.06215902
## 9  120.01955   3.71676939
## 10 130.67097   5.71512130
```

```
## 10 130.67097   5.71512130
## 11 122.39196   3.55995870
## 12  94.81522   2.06440741
## 13   0.00000   0.06666667
## 14 116.76014   3.55831905
## 15 111.77525   2.55107292
```

Filtremos por clusters con mas datos

```
# 1er Cluster con mas datos
data_c1 <- data_pre %>%
  filter(data_pre$clust == 1)

# 2do Cluster con mas datos
data_c2 <- data_pre %>%
  filter(data_pre$clust == 3)

# 3er Cluster con mas datos
data_c3 <- data_pre %>%
  filter(data_pre$clust == 5)
```

Tomemos a c2

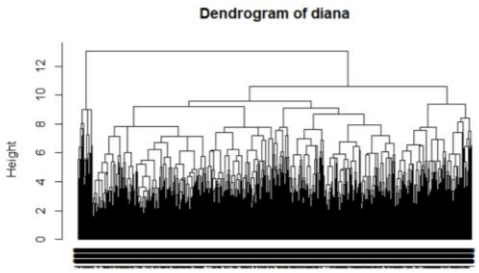
```
# Borramos la columna clust para escalar la datanum de c2
data_c2$clust <- NULL

# Selecciono las variables numericas, se escalan las variables y se almacenan los datos en una tabla
datanumc2 <- data_c2 %>%
  select(data_dou) %>%
  scale() %>%
  as_tibble()
```

Clustering Divisivo

```
# Generamos un modelo divisvo mediante la funcion diana de clusters
modelo_div <- diana(datanumc2)

# Le pedimos el coeficiente de divisivilidad al modelo
modelo_div$dc
```

```
## [1] 0.8340419
```

```
# Graficamos nuestro dendrograma divisivo
pltree(modelo_div, cex = 0.8, hang = -1.5, main = "Dendrogram of diana")
```

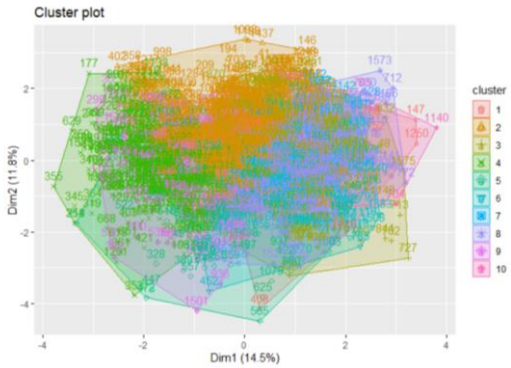


Cantidad Clusters

```
# Para el caso divisivo le entregaremos el numero de clusters con los que queremos agrupar nuestros datos
groupsc2 <- cutree(modelo_div, k = 10)

# Se imprimen los tamaños de cada cluster
table(groupsc2)
```

```
## groupsc2
##   1   2   3   4   5   6   7   8   9  10
##  23 458 207 270  64 142  28 285 101   7
```

```
# Generamos una nueva columna para almacenar a que cluster pertenece cada observacion de data_c2
data_c2$clust <- as.factor(groupsc2)

# Graficamos las observaciones agrupadas por su cluster
fviz_cluster(list(data = datanumc2, cluster = groupsc2))
```




```
# Generamos una nueva columna para almacenar a que cluster pertenece cada observacion de datanumc2
datanumc2$clust <- as.factor(groupsc2)
```

Caracteristicas Clusters encontrados

```
datanumc2$clust <- as.numeric(as.character(datanumc2$clust))

# Generamos una tabla que almacenara los valores promedios para cada uno de los clusters encontrados lo que nos permi
infoclustersc2 <- aggregate(datanumc2, by=list(cluster=datanumc2$clust), mean)

# Borramos la columna clust ya que se repite esa informacion en la tabla generada
infoclustersc2$clust <- NULL

# Transformamos el tiempo de la cancion a minutos
infoclustersc2 <- infoclustersc2 %>% mutate(duration_min = infoclustersc2$duration_ms/60000)

# Borramos la columna de la duracion en milisegundoss
infoclustersc2$duration_ms <- NULL

infoclustersc2
```

```
##    cluster track_popularity danceability     energy         key   loudness
## 1        1       0.20242675   -0.3508700 -0.1786150 -0.51384092 -0.6622772
## 2        2      -0.01747204    0.2212869  0.4619056  0.57241043  0.5315787
## 3        3      -0.30404161   -0.6374021  0.2253350 -0.25910923 -0.4737396
## 4        4       0.72414060    0.6345272 -0.0416548 -0.29307346  0.5736330
## 5        5       0.11108094   -0.5919897 -1.1387957 -0.06464888 -0.9046936
## 6        6      -0.17355226   -0.3048567 -0.8890307 -0.13157254 -1.0765660
## 7        7      -0.38593578   -0.2042990  0.4034304  0.79100916 -0.1474722
## 8        8      -0.52046867   -0.2767870 -0.1516356 -0.41250155 -0.3348126
## 9        9       0.53845241    0.3572377 -0.1318681  0.01796102  0.2845998
```

```
##    cluster track_popularity danceability     energy         key   loudness
## 1        1       0.20242675   -0.3508700 -0.1786150 -0.51384092 -0.6622772
## 2        2      -0.01747204    0.2212869  0.4619056  0.57241043  0.5315787
## 3        3      -0.30404161   -0.6374021  0.2253350 -0.25910923 -0.4737396
## 4        4       0.72414060    0.6345272 -0.0416548 -0.29307346  0.5736330
## 5        5       0.11108094   -0.5919897 -1.1387957 -0.06464888 -0.9046936
## 6        6      -0.17355226   -0.3048567 -0.8890307 -0.13157254 -1.0765660
## 7        7      -0.38593578   -0.2042990  0.4034304  0.79100916 -0.1474722
## 8        8      -0.52046867   -0.2767870 -0.1516356 -0.41250155 -0.3348126
## 9        9       0.53845241    0.3572377 -0.1318681  0.01796102  0.2845998
## 10      10      -0.99194984   -0.4226951  0.2174091 -0.16553433 -0.4954998
##          mode speechiness acousticness instrumentalness    liveness    valence
## 1   0.7370064  -0.3980704  -0.42638972       3.73262740 -0.07222335 -0.1195835
## 2  -0.3642239  -0.1377701   0.01014138      -0.20153352 -0.36908647  0.2107325
## 3   0.3796306  -0.6783254  -0.29640960      -0.15291698  1.35464235  0.2568028
## 4  -0.1425887   1.2111677  -0.06827024      -0.23765191 -0.28266091 -0.4212483
## 5   0.2167892   0.7756147   1.89427292      -0.08614808 -0.37086493 -0.1684778
## 6  -0.8466975  -0.5199859   0.06321594      -0.15204297 -0.48567257 -0.1501757
## 7  -0.9301723  -0.6672084  -0.54643526       5.20318424 -0.11957098 -0.8107106
## 8   0.9557021  -0.6087453  -0.10755817      -0.11046705 -0.54608804  0.1685100
## 9  -0.2493393   1.0422302  -0.08431417      -0.19459141  2.18745562 -0.3195414
## 10 -0.7159538  -0.4628549   1.31597531       4.97471608 -0.37698028  0.8385042
##          tempo  duration_min
## 1  -0.22929159 -2.475931e-06
## 2  -0.22960484 -2.614362e-06
## 3  -0.25515215  4.651857e-06
## 4   0.60420141 -9.159718e-06
## 5   0.98878692 -9.040806e-06
## 6  -0.37769887  1.539231e-05
## 7  -0.07231632 -4.443127e-06
## 8  -0.05623536  4.488449e-06
## 9   0.05562463 -1.389730e-06
## 10  0.41422310  2.042561e-05
```