```
---
title: "Parcial 2"
author: "Sergio Carrero"
date: '2022-06-16'
output:
  pdf_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(plotly)
library(factoextra)
library(dplyr)
library(psych)
install.packages("webshot")
webshot::install_phantomjs()
```

Clase 20 - Clustering

```{r}
dfcie <- read_excel("Cielab_tueste.xlsx")

#View(Cielab_tueste_cafe)
```

```{r}
fig <- plot_ly(data = dfcie,
               x = ~L,
               y = ~a,
               z = ~b,
               size = 0.7,
               color = ~tueste)
fig
```
Se ven nubes de puntos solapadas entre claro, medio y oscuro.

Métodos de agrupamiento (Cluestering)

* K-means (K-medias) para creación de cluster


```{r}
## Número óptimo de clusters
M = dfcie[ ,-4]
Ms = scale(M)
fviz_nbclust(Ms,
             FUNcluster = kmeans,
             method = 'gap_stat',
             diss = get_dist(Ms,
                             'euclidean'))
```

El grafico GAP: A traves de la anterior estandarización son las que me permiten entender el
numero de cluster Ideal que puedo realizar para el conjunto de datos.
```{r}
clus1 = kmeans(Ms, 3)
dfcie$cluster <- clus1$cluster
dfcie <- dfcie[-4]
dfcie |>
  group_by(cluster) |>
  summarise(media_a = mean(a),
            media_b = mean(b),
```
```

```
            media_L = mean(L),
            desv_a = sd(a),
            desv_b = sd(b),
            desv_L = sd(L),
            coeV_a = 100 * desv_a/media_a,
            coeV_b = 100 *desv_b/media_b,
            coeV_L = 100 *desv_L/media_L)

#table(dfcie$tueste,
#      clus1$cluster) ## Matriz de confusión
```

Tpificar: Aca puedo sacar de las coordenadas las 3 medias, desviaciones y coeficiente de variación.

```{r}
# clus1 = kmeans(Ms, 3)
# table(dfcie$tueste,
#      clus1$cluster) ## Matriz de confusión
```

Hice una tabulación de cluster para comparar los tres niveles de tostión.
#Se crea una matriz de confusión, donde el total de datos la tostion clara los 30 datos se van al cluster 3, 28 datos se fueron al cluster 2, y la tostion media no fue tan buena que el claro pero se fueron al grupo 2,
* El grupo 3 (Cluster) representa al tostion 'claro'

* Prácticamente la tostión media pertenece al grupo 2 (Cluster)

* Prácticamente la tostión oscuro pertenece al grupo 2 (Cluster)

** medio y oscuro, pertenece al mismo grupo (Cluster 2) **

* Prácticamente la tostión verde pertenece al grupo 1 (Cluster)

Conclusión, hay 3 Cluster

Método de Ward

```{r}
# Ward Hierarchical Clustering
df_scale <- scale(dfcie) #Se estandariza para que todas las variables esten ajustados.
d <- dist(df_scale, method = "euclidean") # distance matrix
fit <- hclust(d, method="ward.D2")
plot(fit) # display dendogram
groups <- cutree(fit, k=3) # genere un arbol de 3 clusters
# dibujar dendograma con 3 bordes alrededor de 3 clusters
rect.hclust(fit, k=3, border="red")
```

#Dendrograma: Calculando distancias euclideas, aca podemos ver que observación cae en los CLuster.

```{r}
tapply(dfcie$a, groups, mean)
tapply(dfcie$b, groups, mean)
tapply(dfcie$L, groups, mean)
```

#ACA PODEMOS ver que la coordenada mas alta la tiene la coordenada 3.

Asignación para el próximo lunes

* Correr el mismo código de clase, para Chroma y Hue

* Caracterizar los Cluster

* Utilizar algún gráfico que muestre el número de cluster

* Pasar de coordenadas Lab a RGB (Investigar) y hacer los cluster con las coordanadas RGB

* Existe alguna relación del Chroma y Hue para coordenadas RGB, si es cierto, entonces realiza los cluster con RGB

* Pueden colocar otro tipo de coordenadas (ej. HLS, HVS)

#Realización del ejercicio

#Cluster de particiones: No jerarquico cada instancia se coloca exactamente en uno de los K Clusters no superpuestos, solo se tiene un conjunto de conglomerados, el usuario dice el numero deseado de K conglomerados.
#Algotirmo de particion 1: k medias
#1. se Decide el valor de k
#2. Iniciar los k centroides de cluster
#3. Decidir la pertenencia a la clase de los N objetos asignandolos al centroide
#4. Vuelva a estimar los k Centros de luster, asumiendo membresias encontradas
#5. Si ninguno de los N objetos ha cambiado de pertenencia en la ultima iteración, se sale.
Los centroides mutan.
Se debe tener en cuenta que tipo de algoritmo se usa porque se forman grupos pero no necesariamente son los ideales.

#Grafico de sedimentaci{on: Doonde el grafico se vuelve estable donde se forma una recta al final dandome el numero de datos ideales a realiza.r

#Tomamos una muestra de la población para estimar cosas "parametros poblacionales" de la misma población, la media es la media muestral, y la media poblacional o real.
Superpoblacion:

#1 Se hace un muestreo es si la población es finita o infinita, si conozco o no el tamaño de individuos dentro de ella.

#2 Estimar una caracteristica respecto a un total

#3 Es de tipo probabilistico (Misma posibilidades, muestreo irrestricto aleatorio, muestreo sistematico, M.estratificado, M.conglomerados, M.polietapicos) o NO PROBABILISTICO (Sin formulas, Muestreos por cuotas, intencional, M.Bola de nieve, M.Opinatico)

#4 Tecnicas

```{r}
Chroma <- sqrt (dfcie$a^2+dfcie$b^2)
Hue <- atan(dfcie$b/dfcie$a)
CyH <- matrix(c(Chroma,Hue),
              ncol=2)
CyH
```
Número óptimo de clusters
```{r}
G = CyH
Gs = scale(G)
fviz_nbclust(Gs,
             FUNcluster = kmeans,
             method = 'gap_stat',
             diss = get_dist(Gs,
                             'euclidean'))
```
Número óptimo o IDEAL de clusters
```{r}
G = dfcie[ ,-1]
G2 = G [ ,-1]
G3 = G2 [ ,-1]
G4 = G3 [ ,-1]
```

```
Gs = scale(G4)
fviz_nbclust(Gs,
             FUNcluster = kmeans,
             method = 'gap_stat',
             diss = get_dist(Gs,
                             'euclidean'))
```

```{r}
cluster1 = kmeans(Gs, 3)
G4$cluster <- cluster1$cluster
G4 |>
  group_by(cluster) |>
  summarise(mChroma = mean(Chroma),
            mHue = mean(Hue),
            sdChroma = sd(Chroma),
            sdHue = sd(Hue),
            CVChroma = 100 * sdChroma/mChroma,
            CVHue = 100 *sdHue/sdHue,)
```

```{r}
df_scale <- scale(G4)
d <- dist(df_scale, method = "euclidean")
fit <- hclust(d, method="ward.D2")
plot(fit)
groups <- cutree(fit, k=3)
rect.hclust(fit, k=3, border="red")
```

```{r}
# tapply(G4$Chroma, groups, mean)
# tapply(G4$Hue, groups, mean)
```

#Conversion a RGB
```{r}
dfcie1 <- dfcie [ ,-4]
dfcie2 <- dfcie1 [ ,-4]
dfcie3 <- dfcie2 [ ,-4]
dfcie3
library(farver)
dfRGB <- convert_colour(dfcie3,'lab','RGB')
dfRGB
```
Calcular la cantidad de Cluster para RGB
```{r}
MsRGB = scale(dfRGB)
fviz_nbclust(MsRGB,
             FUNcluster = kmeans,
             method = 'gap_stat',
             diss = get_dist(MsRGB,
                             'euclidean'))
```
#lA CANTIDAD IDEAL SON: 3 cluster para RGB
```{r}
cluster1 = kmeans(MsRGB, 3)
dfRGB$cluster <- cluster1$cluster
dfRGB |>
  group_by(cluster) |>
  summarise(mr = mean(V1),
            mg = mean(V2),
            mb = mean(V3),
            sdR = sd(V1),
```

```
            sdG = sd(V2),
            sdB = sd(V3),
            cvR = 100 * sdR/mr,
            cvG = 100 *sdG/mg,
            cvB = 100 *sdB/mb)
```

```{r}
df_scale1 <- scale(dfRGB)
d <- dist(df_scale1, method = "euclidean")
fit <- hclust(d, method="ward.D2")
plot(fit)
groups <- cutree(fit, k=3)
rect.hclust(fit, k=3, border="red")
```

```{r}
tapply(dfRGB$V1, groups, mean)
tapply(dfRGB$V2, groups, mean)
tapply(dfRGB$V3, groups, mean)
```