Segmentación de clientes

Práctica de aprendizaje No Supervisado con K-Means

Álvaro Barrio Hernández (alvaro.barrio.hernandez@gmail.com)

Código: GitHub

La segmentación de clientes es una de las aplicaciones más importantes del aprendizaje no supervisado. Mediante técnicas de agrupación en clústeres, las empresas pueden identificar los distintos segmentos de clientes, lo que les permite dirigirse a la base de usuarios potenciales. En este proyecto de aprendizaje automático, utilizaremos la agrupación en clústeres de K-means, que es el algoritmo esencial para agrupar conjuntos de datos sin etiquetar.

¿Qué es la segmentación de clientes? Es el proceso de división del conjunto de clientes en varios grupos de personas que comparten características relevantes para el marketing, como el género, la edad, los intereses y los hábitos de consumo.

Las empresas que implementan la segmentación tienen la idea de que cada cliente tiene requisitos diferentes y requieren un esfuerzo de marketing específico para abordarlos de manera adecuada. Las empresas tienen como objetivo obtener un enfoque más profundo del cliente al que se dirigen. Por lo tanto, su objetivo debe ser específico y debe adaptarse a los requisitos de todos y cada uno de los clientes. Además, a través de los datos recopilados, las empresas pueden obtener una comprensión más profunda de las preferencias de los clientes, así como los requisitos para descubrir segmentos valiosos que les reportarían el máximo beneficio. De esta manera, pueden diseñar estrategias de sus técnicas de marketing de manera más eficiente y minimizar la posibilidad de riesgo en su inversión.

La técnica de segmentación depende de varios diferenciadores clave que dividen a los clientes en grupos a los que dirigirse. Los datos relacionados con la demografía, la geografía, la situación económica, así como los patrones de comportamiento, juegan un papel crucial en la determinación de la dirección de la empresa para abordar los distintos segmentos.

Comencemos por instalar los paquetes necesarios para la correcta realización del ejercicio:

```
library(umap)
library(ggplot2)
library(plotrix)
library(purrr)
library(cluster)
library(gridExtra)
library(grid)
library(NbClust)
library(factoextra)
library(tinytex)
library(tidyverse)
library(readr)
library(Rtsne)
```

1. Datos y características

Comenzamos por la carga de datos desde un archivo csv:

```
customer_data=read.csv("Mall_Customers.csv")
```

Chequeamos la información básica del dataset:

```
head(customer_data)
```

```
##   CustomerID Gender Age Annual.Income..k.. Spending.Score..1.100.
## 1          1   Male  19                 15                     39
## 2          2   Male  21                 15                     81
## 3          3 Female  20                 16                      6
## 4          4 Female  23                 16                     77
## 5          5 Female  31                 17                     40
## 6          6 Female  22                 17                     76
```

```
names(customer_data)
```

```
## [1] "CustomerID"             "Gender"                 "Age"
## [4] "Annual.Income..k.."     "Spending.Score..1.100."
```

```
str(customer_data)
```

```
## 'data.frame':    200 obs. of  5 variables:
##  $ CustomerID            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender                : chr  "Male" "Male" "Female" "Female" ...
##  $ Age                   : int  19 21 20 23 31 22 35 23 64 30 ...
##  $ Annual.Income..k..    : int  15 15 16 16 17 17 18 18 19 19 ...
##  $ Spending.Score..1.100.: int  39 81 6 77 40 76 6 94 3 72 ...
```

```
summary(customer_data)
```

```
##    CustomerID        Gender               Age
Annual.Income..k..
##  Min.   :  1.00   Length:200         Min.   :18.00   Min.   : 15.00
##  1st Qu.: 50.75   Class :character   1st Qu.:28.75   1st Qu.: 41.50
##  Median :100.50   Mode  :character   Median :36.00   Median : 61.50
##  Mean   :100.50                      Mean   :38.85   Mean   : 60.56
##  3rd Qu.:150.25                      3rd Qu.:49.00   3rd Qu.: 78.00
##  Max.   :200.00                      Max.   :70.00   Max.   :137.00
##  Spending.Score..1.100.
##  Min.   : 1.00
##  1st Qu.:34.75
##  Median :50.00
##  Mean   :50.20
##  3rd Qu.:73.00
##  Max.   :99.00
```

Observamos como la edad de nuestros clientes tiene su mínimo en 18 y máximo en 70.

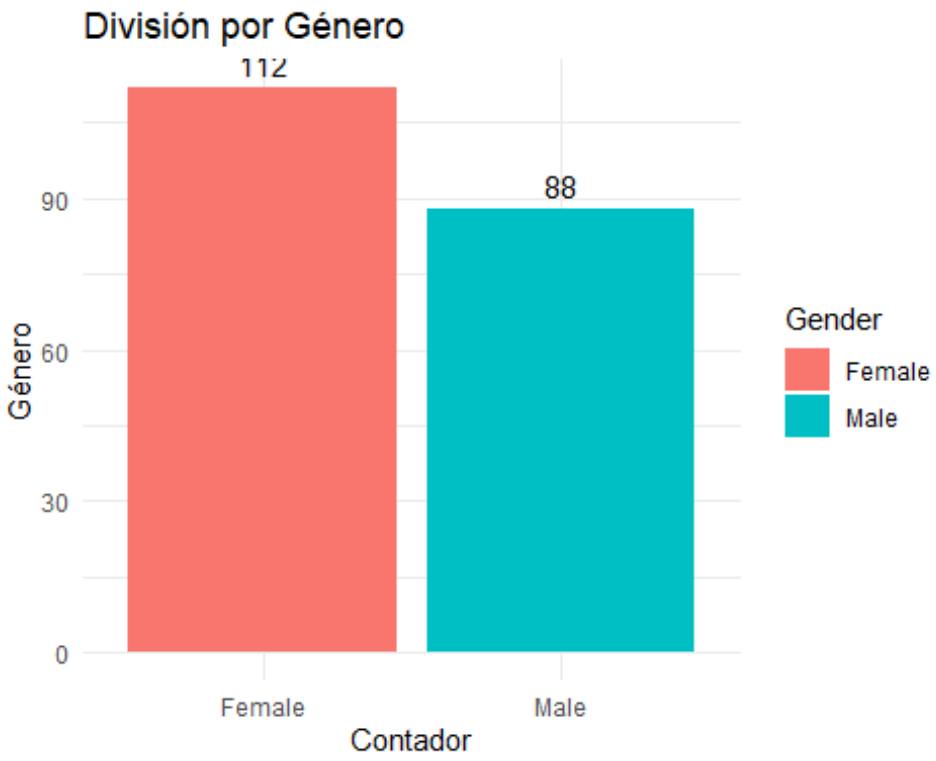
El ‘Anual Income’ comienza en 15.000 y termina en 137.000.

El ‘Spending Score’ se distribuye en el intervalo 1 a 99.

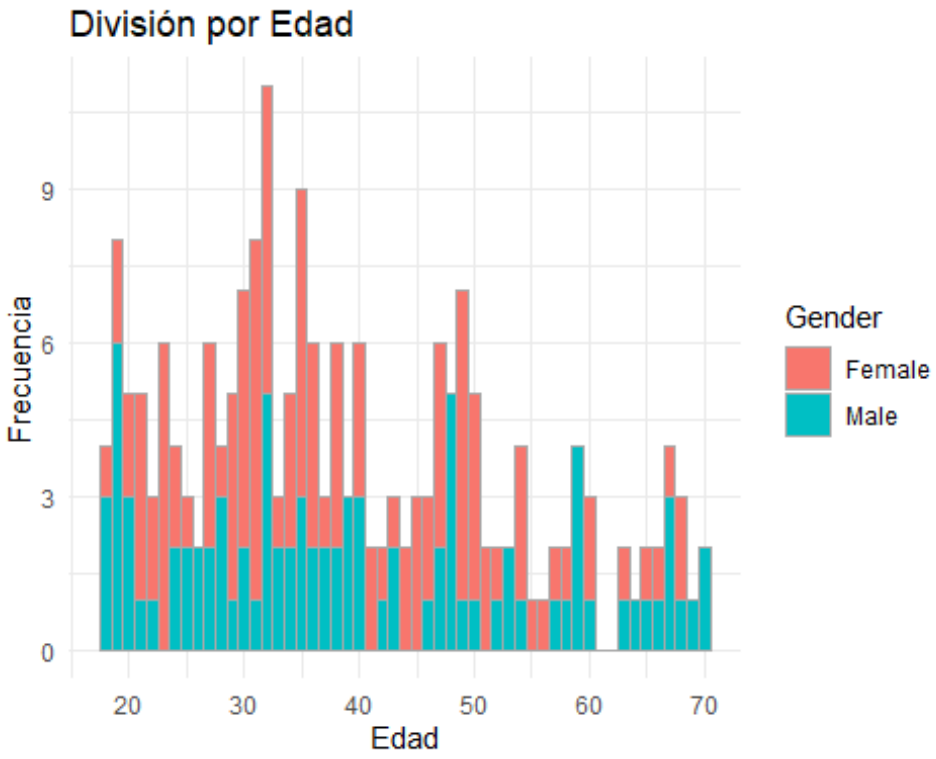
---

2. Visualizaciones

En este apartado se plantean diversas figuras para graficar el comportamiento de cada variable:

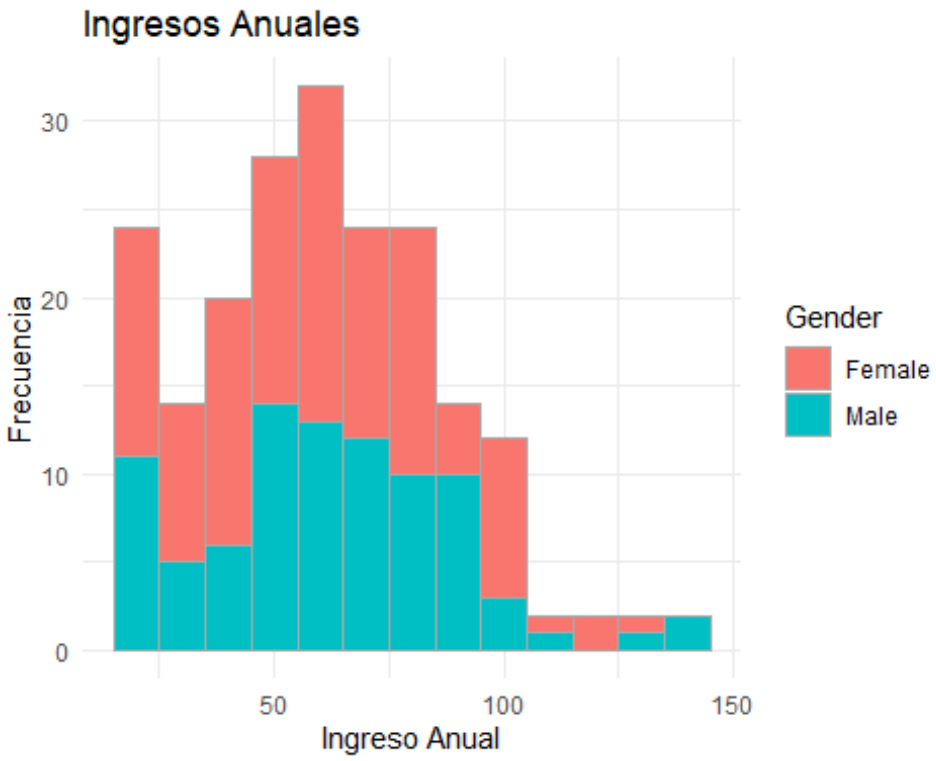


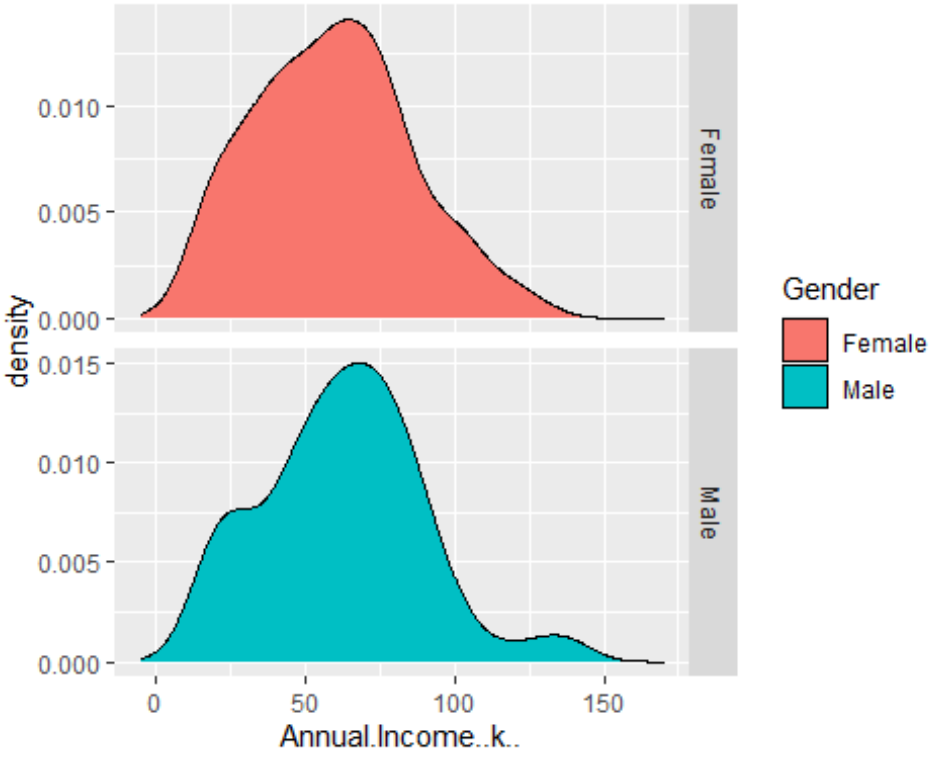
El conjunto de datos contiene un número mayor de registros femeninos frente a masculinos.



Se comprueba como la franja de 30 a 35 años contiene a un gran número de clientes.

El más joven tiene 18 y el mayor 70.

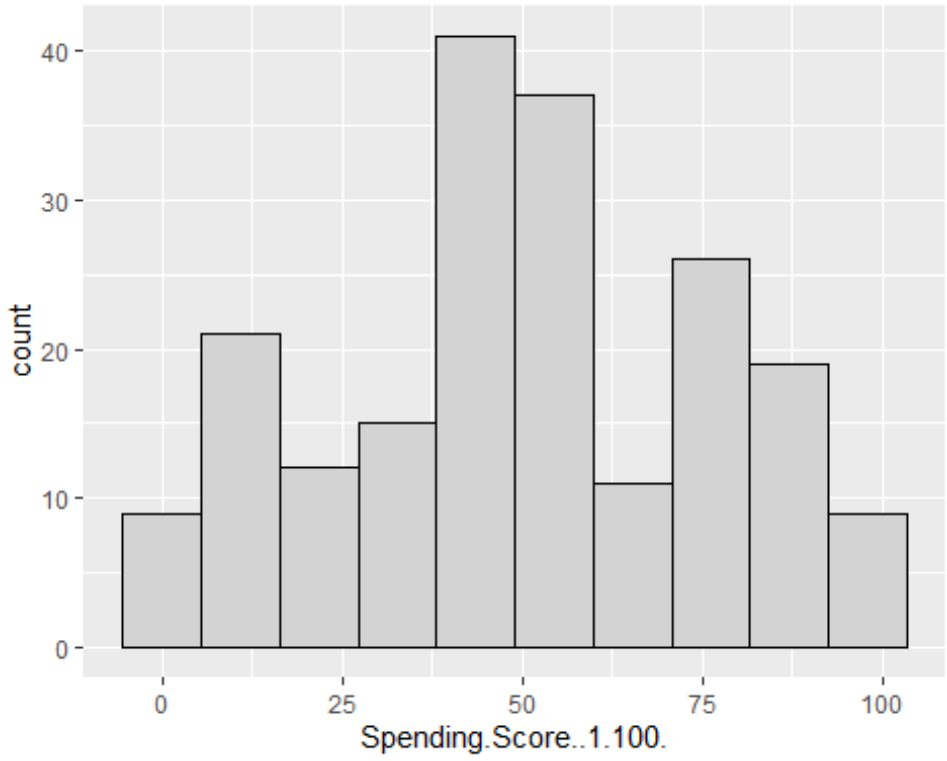




Corroboramos que el ingreso mínimo es 15.000 y máximo 137.000.

Se observa una distribución normal.


Boxplot para Spending Score
0.4
0.2
0.0
-0.2
-0.4
0
25
50
75
100
Age


40
30
20
10
0
count
0
25
50
75
100
Spending.Score..1.100.

Referente al 'Spending Score', el mínimo es 1, el máximo es 99 y la media es 50.

Los clientes entre 40 y 50 años son los que obtienen mayor puntuación en gasto.

Nótese la presencia de agrupaciones naturales.

---

3. Aplicación de Algoritmo de k-Means

1. Para empezar, primero seleccionamos un número de clústeres para usar e inicializamos aleatoriamente sus respectivos puntos centrales. Para calcular el número de clústeres a utilizar, es bueno echar un vistazo rápido a los datos y tratar de identificar cualquier agrupación distinta. Los puntos centrales son vectores de la misma longitud que cada vector de puntos de datos.

2. Cada punto de datos se clasifica calculando la distancia entre ese punto y cada centro del clúster, y luego clasificando el punto que estará en el clúster cuyo centro está más cerca de él.

3. Basándonos en estos puntos clasificados, recalculamos el centro del clúster tomando la media de todos los vectores del clúster.

4. Se repiten estos pasos para un número determinado de iteraciones o hasta que los centros de clústeres no cambien mucho entre iteraciones. También puedes optar por inicializar aleatoriamente los centros de grupo unas cuantas veces, y

luego seleccionar la ejecución que parezca que proporcionó los mejores resultados.

K-Means tiene la ventaja de que es bastante rápido, ya que todo lo que estamos haciendo es calcular las distancias entre puntos y centros de grupo, por lo tanto, son muy pocos cálculos.

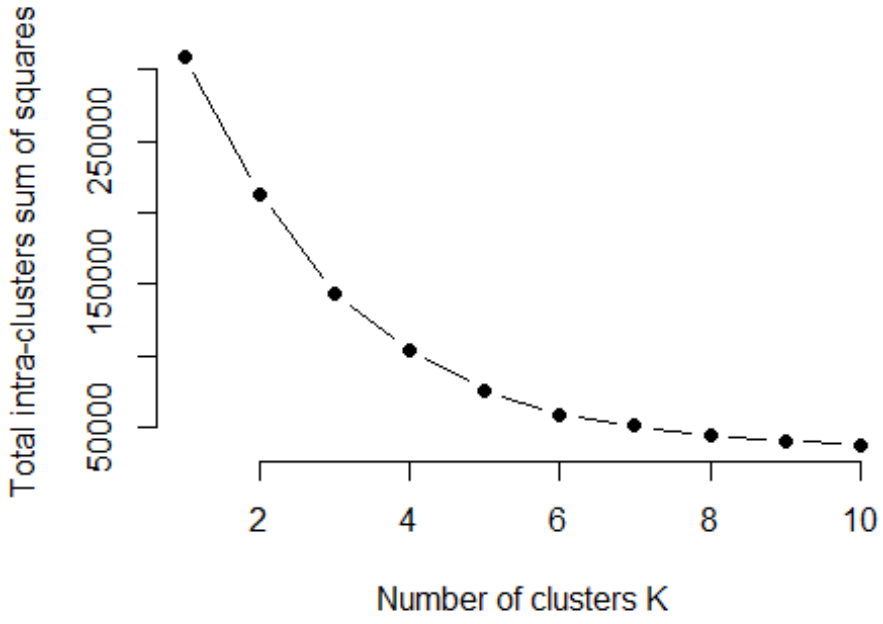
Debemos especificar el número de clústers a utilizar.

Se presentan 3 métodos para ayudar en la selección:

1) Elbow Method

Probablemente el método más conocido. Se buscamos un cambio de pendiente, un codo, para determinar el número óptimo de clústeres. Es un método inexacto, pero sigue siendo potencialmente útil.

Este método funciona de la siguiente forma, se calcula la suma de errores cuadráticos dentro del clúster para diferentes valores de K y se elige la K para la cual la suma de errores cuadráticos comienza a disminuir. Esto es visible como un codo:



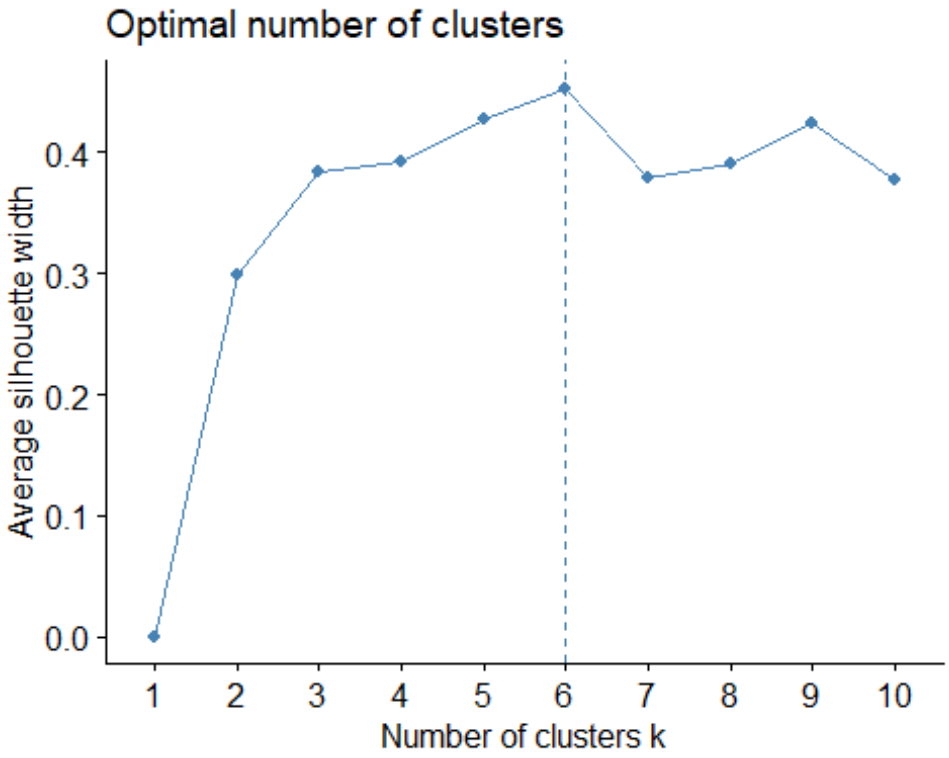
2) Método de la silueta

El método de la silueta puede utilizarse para estudiar la distancia de separación entre los grupos resultantes. Muestra la cercanía entre cada punto en un clúster de los

puntos en los clústeres vecinos. Este método es mejor ya que hace que la decisión sobre el número óptimo de clústeres sea más significativa y clara.

Pero esta métrica es costosa de calcular ya que el coeficiente se calcula para cada caso. Por lo tanto, la decisión sobre la métrica óptima a elegir para el número de clústeres se debe tomar de acuerdo con las necesidades del producto.

Esta medida tiene un rango de -1 a 1. Donde 1 significa que los puntos están muy cerca de su propio clúster y lejos de otros clústeres, mientras que -1 indica que los puntos están cerca de los clústeres vecinos:

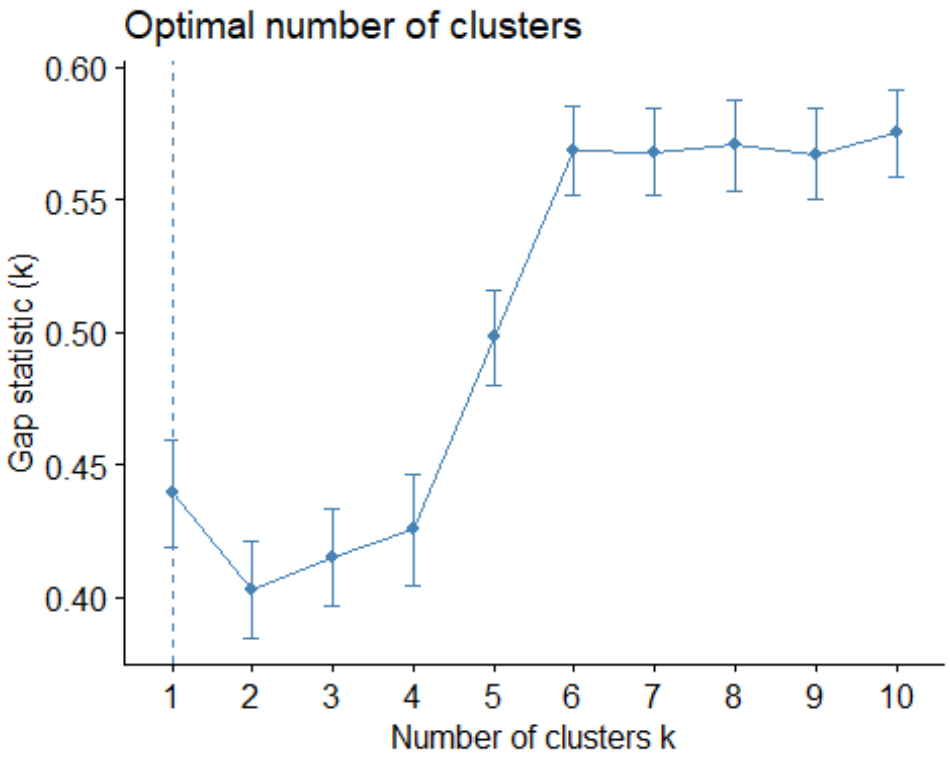


3) Método de estadística de brecha

Este método compara el total dentro de la variación intraclúster para diferentes valores de K con sus valores esperados bajo una distribución de referencia nula de los datos.

La estimación de los clustering óptimos será un valor que maximice la estadística de la brecha, es decir, que produzca la estadística de la brecha más grande. Esto significa que la estructura de agrupación está muy lejos de la distribución uniforme y aleatoria de los puntos.

El gráfico de estadísticas de brecha muestra las estadísticas por número de clústeres con errores estándar dibujados con segmentos verticales y el valor óptimo de K marcado con una línea azul discontinua vertical.



Después de probar los 3 métodos, elegimos k = 6 como número de clústers:

```
## K-means clustering with 6 clusters of sizes 45, 22, 21, 38, 35, 39
##
## Cluster means:
##        Age Annual.Income..k.. Spending.Score..1.100.
## 1 56.15556           53.37778                49.08889
## 2 25.27273           25.72727                79.36364
## 3 44.14286           25.14286                19.52381
## 4 27.00000           56.65789                49.13158
## 5 41.68571           88.22857                17.28571
## 6 32.69231           86.53846                82.12821
##
## Clustering vector:
##   [1] 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
3 2 3 2 3
##  [38] 2 3 2 1 2 1 4 3 2 1 4 4 4 1 4 4 1 1 1 1 1 4 1 1 4 1 1 1 4 1 1 4
4 1 1 1 1
##  [75] 1 4 1 4 4 1 1 4 1 1 4 1 1 4 4 1 1 4 1 4 4 4 1 4 1 4 4 1 1 4 1 4
1 1 1 1 1
## [112] 4 4 4 4 4 1 1 1 1 4 4 4 6 4 6 5 6 5 6 5 6 4 6 5 6 5 6 5 6 5 6 4
6 5 6 5 6
## [149] 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6
5 6 5 6 5
## [186] 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6
```

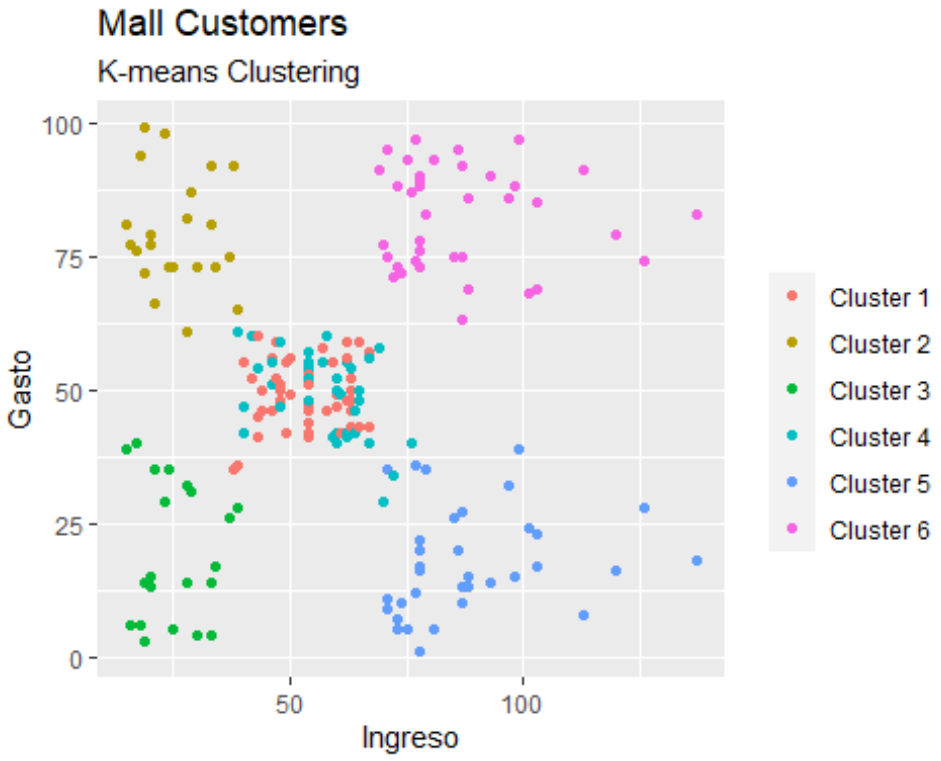
```
##
## Within cluster sum of squares by cluster:
## [1]  8062.133  4099.818  7732.381  7742.895 16690.857 13972.359
##  (between_SS / total_SS =  81.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

---

4. Visualización de Clústers

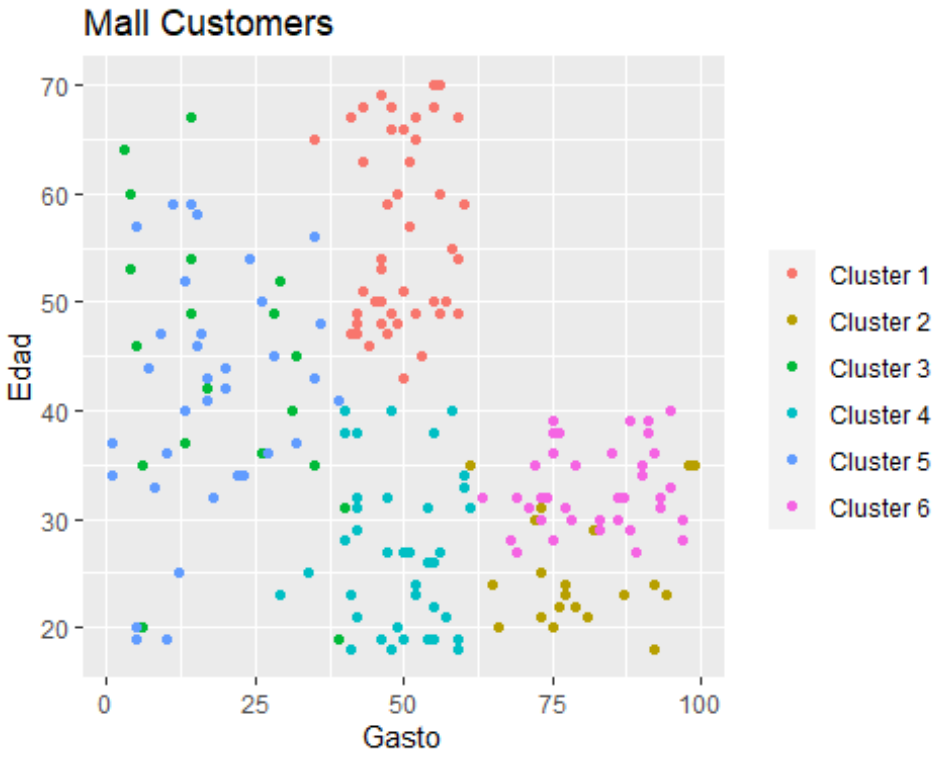
```
## Importance of components:
##                             PC1     PC2     PC3
## Standard deviation     26.4625 26.1597 12.9317
## Proportion of Variance  0.4512  0.4410  0.1078
## Cumulative Proportion   0.4512  0.8922  1.0000

##                                PC1        PC2
## Age                      0.1889742 -0.1309652
## Annual.Income..k..      -0.5886410 -0.8083757
## Spending.Score..1.100. -0.7859965  0.5739136
```



Podemos observar una clasificación de clúster de la siguiente forma:

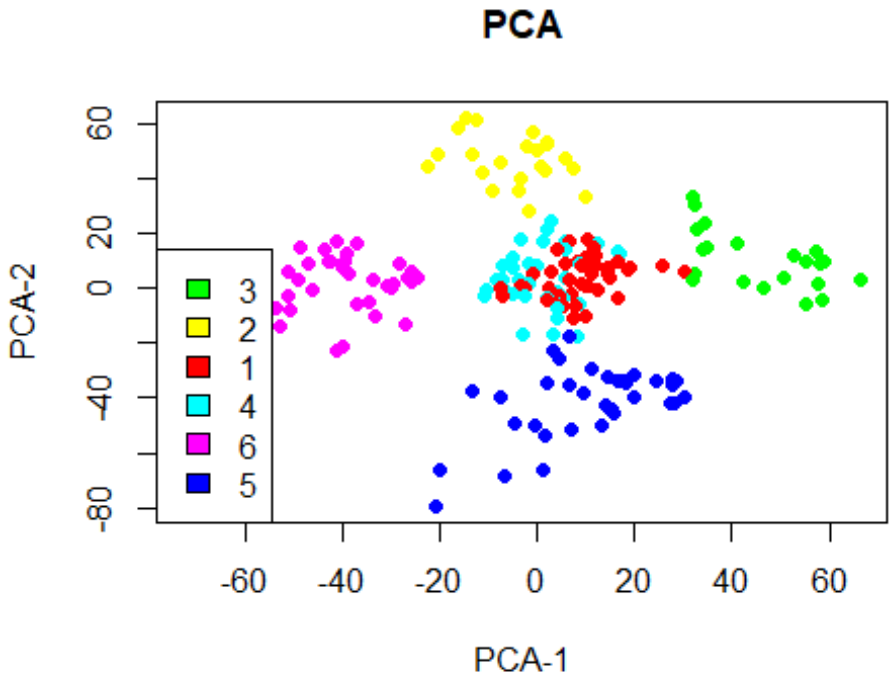
- Clúster 1 y 2 – Ingreso medio y gasto medio
- Clúster 3 – Ingreso alto y gasto alto
- Clúster 4 – Ingreso bajo y gasto bajo
- Clúster 5 – Ingreso bajo y gasto alto
- Clúster 6 – Ingreso alto y gasto bajo



Podemos observar una clasificación de clúster de la siguiente forma:

- Clúster 1 - Gasto medio y edad alta
- Clúster 2 – Gasto medio y edad baja
- Clúster 3 y 5 – Gasto alto y edad baja
- Clúster 4 y 6 – Gasto bajo

5. Principal Components Analysis (PCA) & t-distributed stochastic neighbor embedding (t-sne)



Distribución:

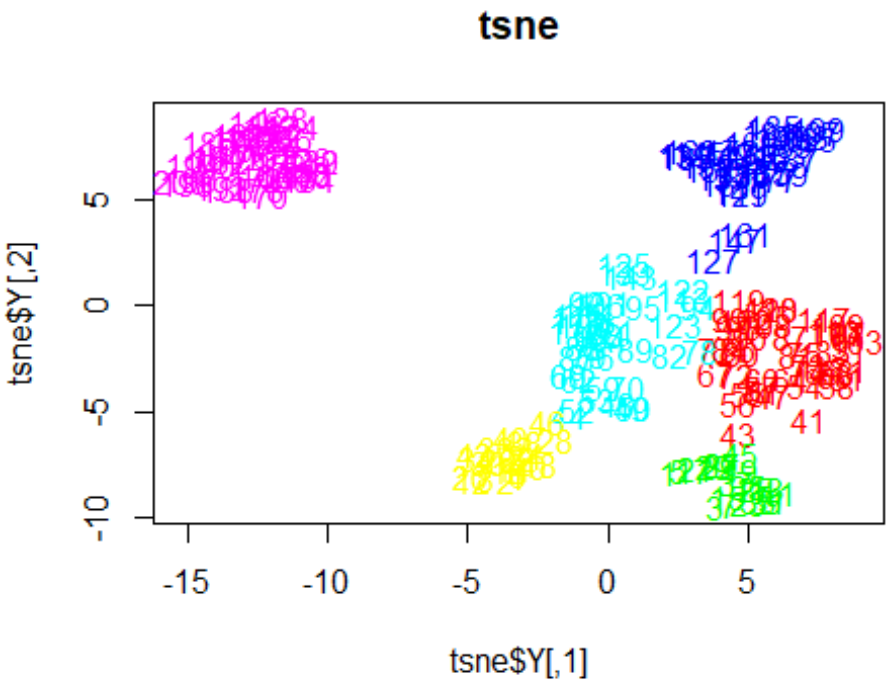
- Clúster 4 y 1: estos dos clústeres están formados por clientes con puntuación PCA1 media y PCA2 media.
- Clúster 6: este clúster representa a los clientes que tienen un PCA2 alto y un PCA1 bajo.
- Clúster 5: en este grupo, hay clientes con una puntuación de PCA1 media y una puntuación de PCA2 baja.
- Clúster 3: este clúster se compone de clientes con un alto ingreso de PCA1 y un alto PCA2.
- Clúster 2: se compone de clientes con un PCA2 alto y un gasto anual medio de ingresos.

```
## Performing PCA
## Read the 200 x 3 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.04 seconds (sparsity = 0.591750)!
## Learning embedding...
## Iteration 50: error is 48.900006 (50 iterations in 0.04 seconds)
```

```
## Iteration 100: error is 47.943164 (50 iterations in 0.04 seconds)
## Iteration 150: error is 47.773249 (50 iterations in 0.03 seconds)
## Iteration 200: error is 47.792989 (50 iterations in 0.03 seconds)
## Iteration 250: error is 47.933106 (50 iterations in 0.03 seconds)
## Iteration 300: error is 0.208232 (50 iterations in 0.02 seconds)
## Iteration 350: error is 0.197127 (50 iterations in 0.02 seconds)
## Iteration 400: error is 0.192847 (50 iterations in 0.03 seconds)
## Iteration 450: error is 0.192261 (50 iterations in 0.02 seconds)
## Iteration 500: error is 0.190709 (50 iterations in 0.03 seconds)
## Fitting performed in 0.29 seconds.

##                      Length Class  Mode
## N                      1    -none- numeric
## Y                    400    -none- numeric
## costs                200    -none- numeric
## itercosts             10    -none- numeric
## origD                  1    -none- numeric
## perplexity             1    -none- numeric
## theta                  1    -none- numeric
## max_iter               1    -none- numeric
## stop_lying_iter        1    -none- numeric
## mom_switch_iter        1    -none- numeric
## momentum               1    -none- numeric
## final_momentum         1    -none- numeric
## eta                    1    -none- numeric
## exaggeration_factor    1    -none- numeric

## Performing PCA
## Read the 200 x 3 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.04 seconds (sparsity = 0.591750)!
## Learning embedding...
## Iteration 50: error is 48.738454 (50 iterations in 0.04 seconds)
## Iteration 100: error is 47.774884 (50 iterations in 0.05 seconds)
## Iteration 150: error is 47.773019 (50 iterations in 0.03 seconds)
## Iteration 200: error is 47.762276 (50 iterations in 0.04 seconds)
## Iteration 250: error is 47.815661 (50 iterations in 0.04 seconds)
## Iteration 300: error is 0.262287 (50 iterations in 0.03 seconds)
## Iteration 350: error is 0.201107 (50 iterations in 0.03 seconds)
## Iteration 400: error is 0.196134 (50 iterations in 0.03 seconds)
## Iteration 450: error is 0.193176 (50 iterations in 0.03 seconds)
## Iteration 500: error is 0.193920 (50 iterations in 0.03 seconds)
## Fitting performed in 0.35 seconds.
```



Con la ayuda de la agrupación en clústeres, podemos comprender mucho mejor las variables, lo que nos impulsa a tomar decisiones cuidadosas. Con la identificación de clientes, las empresas pueden lanzar productos y servicios que se dirigen de forma diferente en función de varios parámetros como ingresos, edad, patrones de gasto, etc. Además, se toman en consideración patrones más complejos como revisiones de productos para una mejor segmentación.

Concluimos la práctica de aprendizaje no supervisado haciendo uso de un algoritmo de agrupamiento llamado K-Means. Analizamos y visualizamos los datos y luego procedimos a implementar nuestro algoritmo.

Pueden encontrar todo el código disponible en mi GitHub

¡Saludos!