Minería de datos para grandes volúmenes de Información

Julián Castelblanco, Cristian David Muñoz

18 de abril de 2020

Resumen

Este documento contiene los lineamientos para la presentación de la última actividad evaluativa a entregar el dia 18 de Abril de 2020.

1. Introducción

El objetivo del proyecto es proveer elementos teóricos y conceptuales que permitan a las empresas entender, y enfrentar el problema de segementar a sus clientes con un modelo compacto que permita representar fenómenos del mundo real.

Consecuentemente, los principios de teoría de aprendizaje fueron adelantados en la primera sesión. Se explorará la construcción de diversos modelos mediante la aplicación de los conceptos. Teniendo en cuenta que los lectores de la Maestría en Ciencia de los Datos y Analítica tienen diferentes perfiles y fundamentación, el proyecto pretende adelantar la correcta aplicación de conceptos que permiten construir el modelo, evaluarlo y juzgar con perspectiva científica su desempeño. Esta actividad evaluativa consiste en aplicar los conceptos de aprendizaje no supervisado en un conjunto de datos.

Los algoritmos que van a explorar son los siguientes:

- K means clustering
- K Medoids clustering
- Fuzzy c-means clustering
- Hierarchical clustering

2. Desarrollo

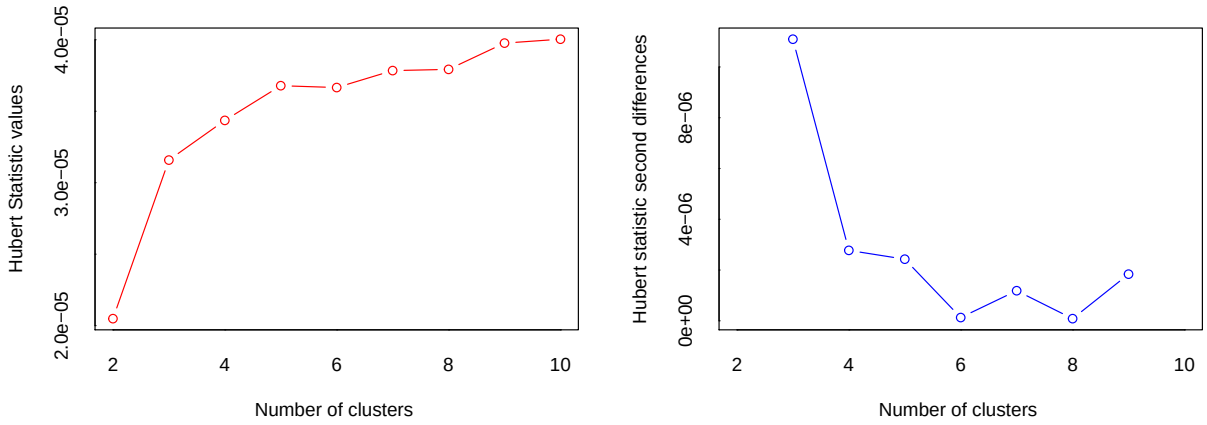
La base de datos seleccionada es **data_ventas**, con la cuál realizaremos los siguientes pasos:

2.1 Número de Clúster óptimo

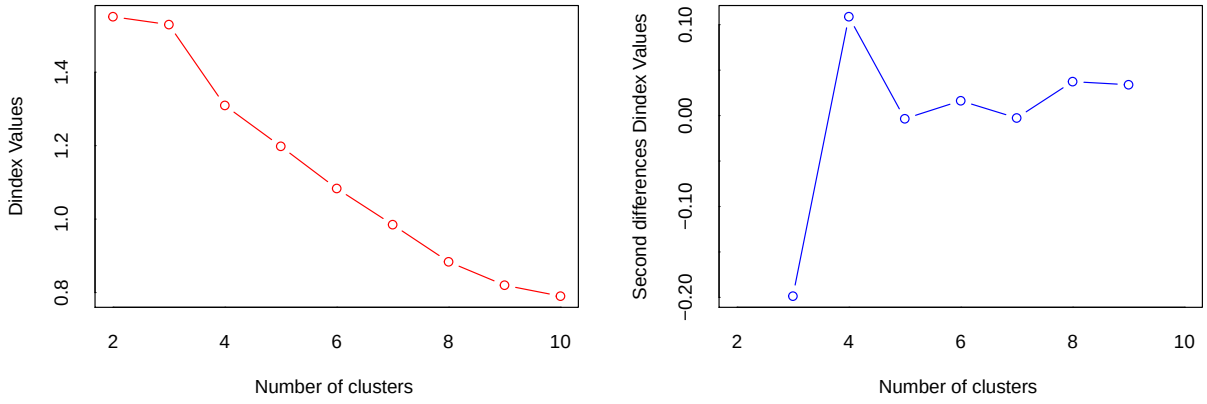
Para encontrar el número de clúster óptimo, realizamos el pico significativo de Hubert *1*, con los siguientes criterios:

- **Distancia:** *Manhattan*, *Euclidean* y *Canberra*.
  - *Manhattan*: $d(x, y) = sum_{j=1}^{d}|x_j - y_j|$
- **Método:**
  - *Ward*, el método minimiza el número total de clusters respecto a la varianza.
- **Índice:** *all*, todos los índeces excepto GAP, Gamma, Gplus y Tau

Manhattan y Ward



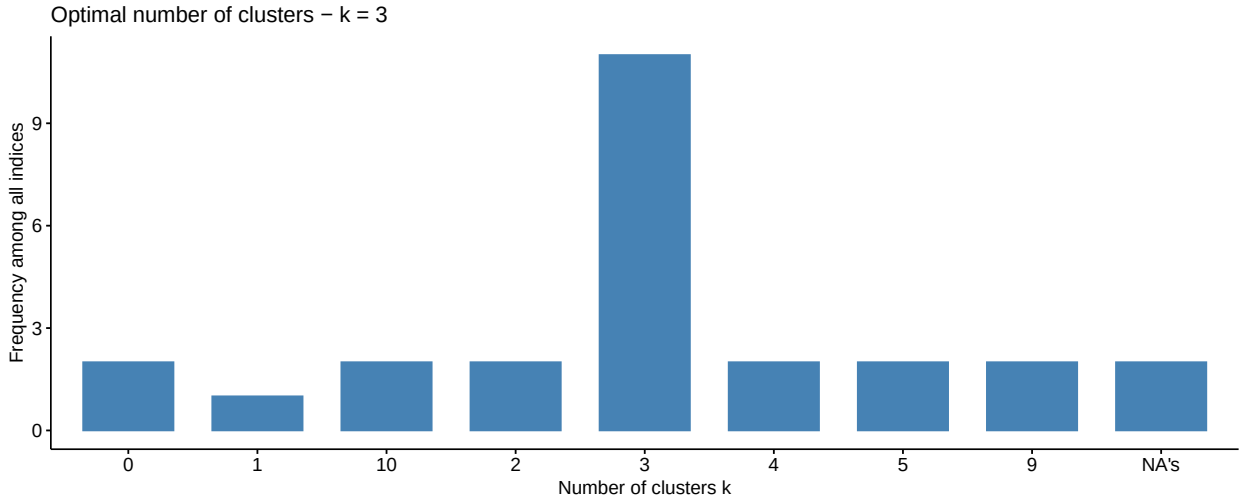

```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 11 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 2 proposed 9 as the best number of clusters
## * 2 proposed 10 as the best number of clusters
##
##                   ***** Conclusion *****
```

```
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 2 proposed  2 as the best number of clusters
## * 11 proposed  3 as the best number of clusters
## * 2 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 2 proposed  9 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
## * 2 proposed  NA's as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  3 .
```

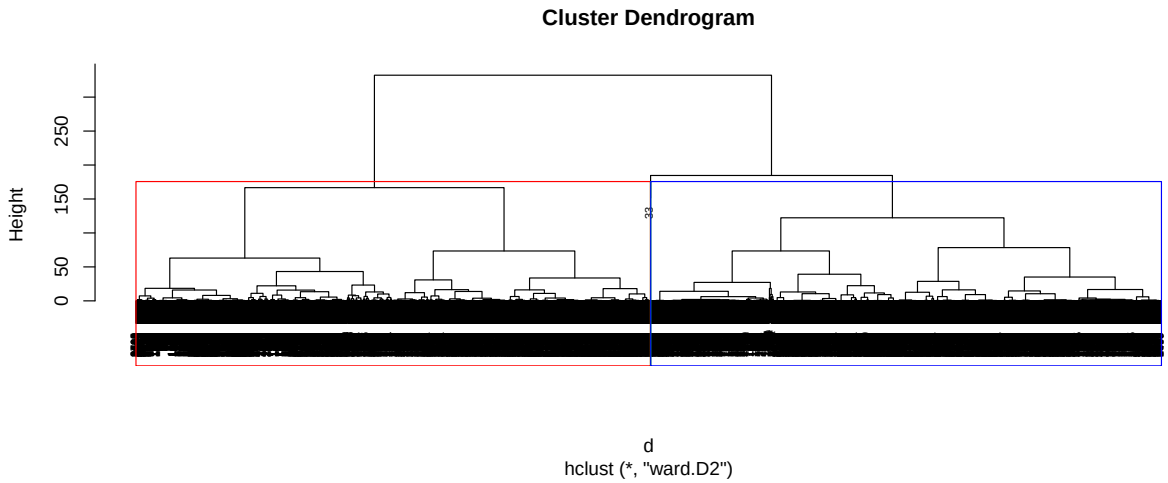


Hierarchical Clustering

Es una alternativa a los métodos de *partitioning clustering* que no requiere que se pre-especifique el número de clusters. Los métodos que engloba el **hierarchical clustering** *7* se subdividen en dos tipos dependiendo de la estrategia seguida para crear los grupos:

Agglomerative clustering (bottom-up) y Divisive clustering (top-down). En ambos casos, los resultados pueden representarse de forma muy intuitiva en una estructura de árbol llamada ***dendrograma***.

Manhattan y Ward



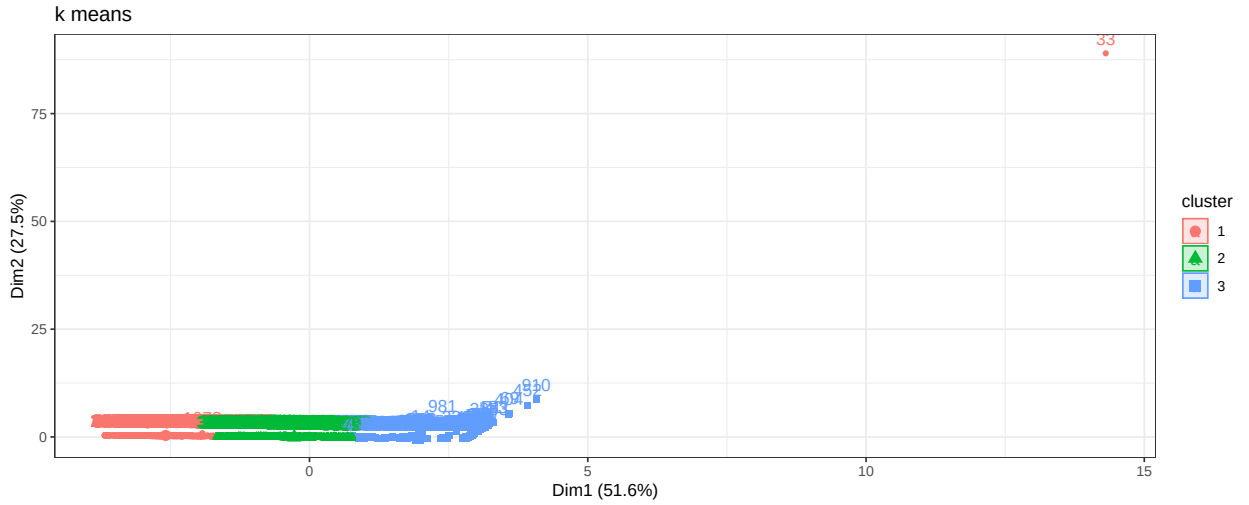
Para nuestro caso elegimos el método **Ward** con distancia **Manhattan** ya que son los criterios más usados en los métodos de clusterización, principalmente porque minimiza la variabilidad *(Ward)* y la distancia *(Manhattan)* se ve menos afectada por outliers (es más robusta) que la distancia euclídea debido a que no eleva al cuadrado las diferencias. Así mismo para la realización de los modelos, se eligen **3** como el númer clúster óptimo por el índice de Hubert; los métodos a realizar son los siguientes:

- K means clustering *2*
- K medoids *3*
- Fuzzy c-means clustering *4*
- Clara Clustering *5*

2.2 Clusterización en Altas dimensiones

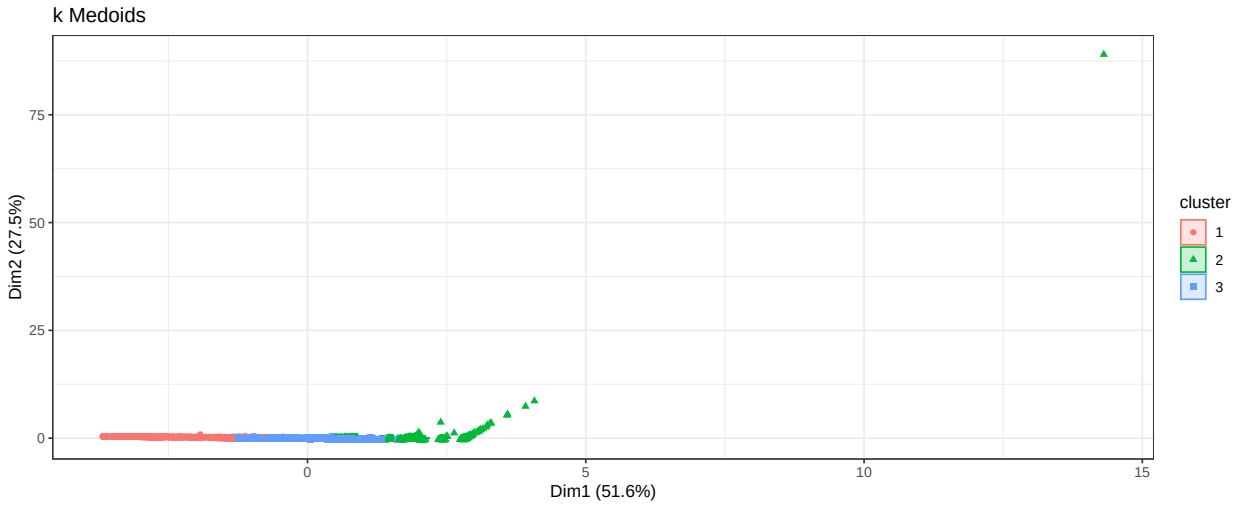
2.2.1 K means clustering

El método K-means clustering (MacQueen, 1967) agrupa las observaciones en K clusters distintos, donde el número K lo determina el analista antes de ejecutar del algoritmo. K-means clustering encuentra los K mejores clusters, entendiendo como mejor cluster aquel cuya varianza interna (intra-cluster variation) sea lo más pequeña posible. Se trata por lo tanto de un problema de optimización, en el que se reparten las observaciones en K clusters de forma que la suma de las varianzas internas de todos ellos sea lo menor posible.



2.2.2 K medoids

K-medoids es un método de clustering donde el elemento dentro de un cluster cuya distancia (diferencia) promedio entre él y todos los demás elementos del mismo cluster es lo menor posible. Se corresponde con el elemento más central del cluster y por lo tanto puede considerarse como el más representativo. El hecho de utilizar medoids en lugar de centroides hace de K-medoids un método más robusto que K-means, viéndose menos afectado por outliers o ruido. A modo de idea intuitiva puede considerarse como la analogía entre media y mediana.

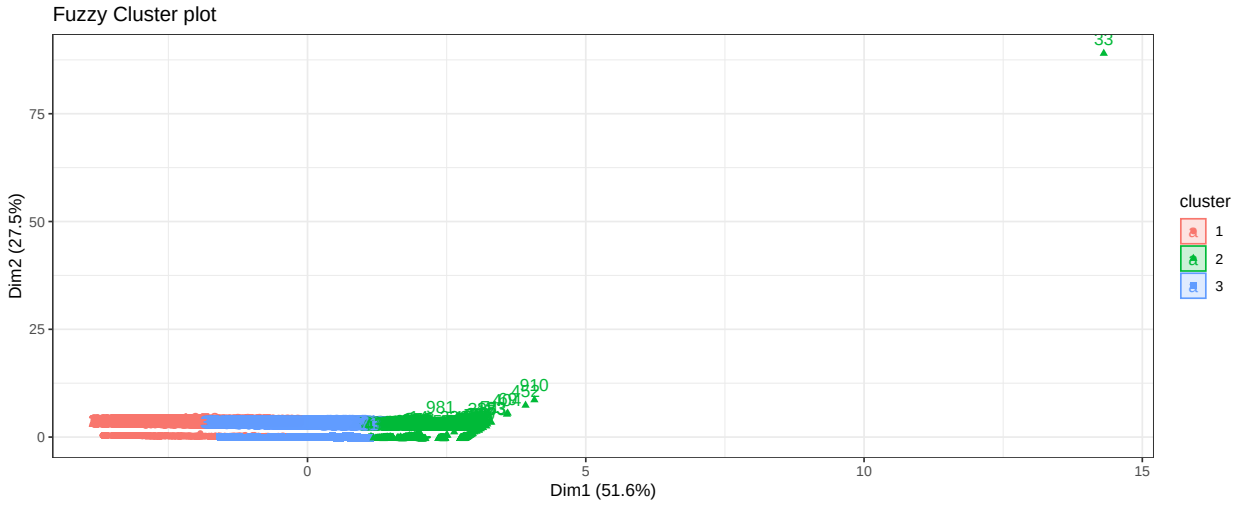


2.2.3 **Fuzzy c-means clustering**

Los métodos de fuzzy clustering o soft clustering se caracterizan porque, cada observación, puede pertenecer potencialmente a varios clusters, en concreto, cada observación tiene asignado un grado de pertenencia a cada uno de los cluster.

Se asemeja en gran medida al algoritmo de k-means pero con dos diferencias:

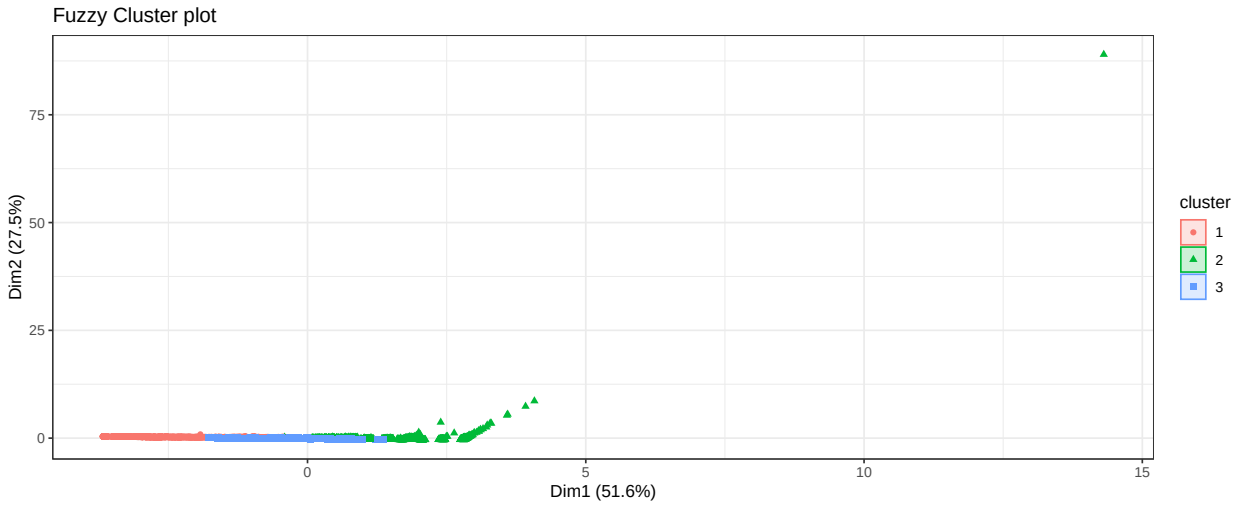
- El cálculo de los centroides de los clusters. La definición de centroide empleada por c-means es: la media de todas las observaciones del set de datos ponderada por la probabilidad de pertenecer a al cluster.
- Devuelve para cada observación la probabilidad de pertenecer a cada cluster.



2.2.4 **Clara Clustering**

Jerárquico es un método que selecciona una muestra aleatoria de un tamaño determinado y le aplica el

algoritmo de PAM (K-medoids) para encontrar los clusters óptimos acorde a esa muestra. Utilizando esos medoids se agrupan las observaciones de todo el set de datos. La calidad de los medoids resultantes se cuantifica con la suma total de las distancias entre cada observación del set de datos y su correspondiente medoid (suma total de distancias intra-clusters). CLARA repite este proceso un número predeterminado de veces con el objetivo de reducir el tiempo de muestreo. Por último, se seleccionan como clusters finales los obtenidos con aquellos medoids que han conseguido menor suma total de distancias.



2.3 Clusterización con Embebimiento TSNE

T-SNE 8 es un método útil de reducción de dimensionalidad que le permite visualizar datos incrustados en un número menor de dimensiones. Puede lidiar con patrones más complejos de grupos gaussianos en el espacio multidimensional en comparación con *PCA*.

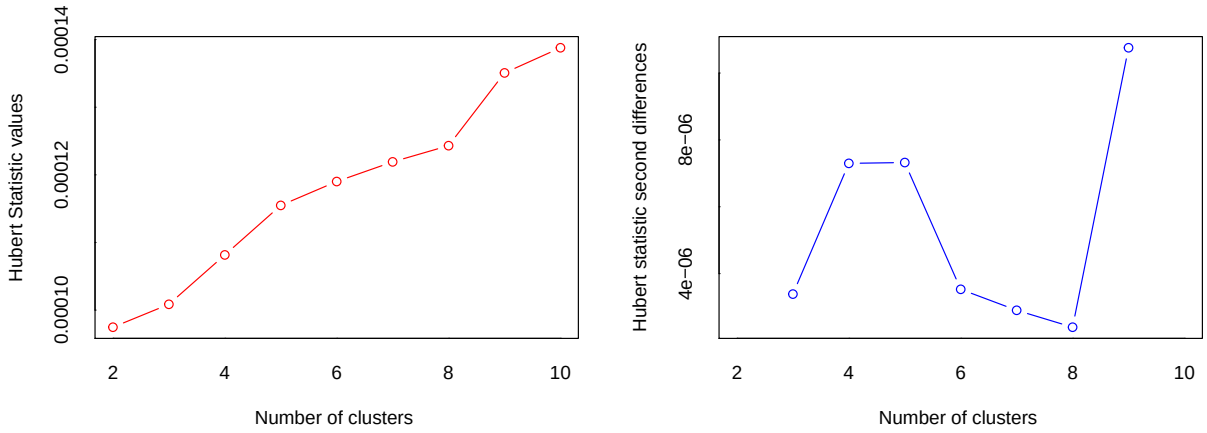
```
## Performing PCA
## Read the 4373 x 7 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 3, perplexity = 100.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 3.81 seconds (sparsity = 0.082281)!
## Learning embedding...
## Iteration 50: error is 73.310072 (50 iterations in 1.73 seconds)
## Iteration 100: error is 55.384110 (50 iterations in 1.72 seconds)
## Iteration 150: error is 53.679764 (50 iterations in 1.59 seconds)
## Iteration 200: error is 53.113107 (50 iterations in 1.65 seconds)
## Iteration 250: error is 52.843592 (50 iterations in 1.68 seconds)
## Iteration 300: error is 0.859984 (50 iterations in 1.49 seconds)
## Iteration 350: error is 0.609971 (50 iterations in 1.46 seconds)
## Iteration 400: error is 0.504344 (50 iterations in 1.41 seconds)
## Iteration 450: error is 0.451794 (50 iterations in 1.39 seconds)
## Iteration 500: error is 0.423035 (50 iterations in 1.38 seconds)
## Iteration 550: error is 0.407372 (50 iterations in 1.38 seconds)
## Iteration 600: error is 0.397570 (50 iterations in 1.36 seconds)
## Iteration 650: error is 0.389696 (50 iterations in 1.35 seconds)
## Iteration 700: error is 0.382874 (50 iterations in 1.32 seconds)
## Iteration 750: error is 0.378816 (50 iterations in 1.30 seconds)
## Iteration 800: error is 0.375639 (50 iterations in 1.29 seconds)
## Iteration 850: error is 0.371628 (50 iterations in 1.28 seconds)
```

```
## Iteration 900: error is 0.368373 (50 iterations in 1.29 seconds)
## Iteration 950: error is 0.366183 (50 iterations in 1.31 seconds)
## Iteration 1000: error is 0.363730 (50 iterations in 1.31 seconds)
## Iteration 1050: error is 0.361355 (50 iterations in 1.31 seconds)
## Iteration 1100: error is 0.359989 (50 iterations in 1.31 seconds)
## Iteration 1150: error is 0.358749 (50 iterations in 1.30 seconds)
## Iteration 1200: error is 0.357655 (50 iterations in 1.32 seconds)
## Iteration 1250: error is 0.356096 (50 iterations in 1.32 seconds)
## Iteration 1300: error is 0.355059 (50 iterations in 1.31 seconds)
## Iteration 1350: error is 0.354198 (50 iterations in 1.32 seconds)
## Iteration 1400: error is 0.353087 (50 iterations in 1.30 seconds)
## Iteration 1450: error is 0.352543 (50 iterations in 1.30 seconds)
## Iteration 1500: error is 0.351664 (50 iterations in 1.30 seconds)
## Iteration 1550: error is 0.350980 (50 iterations in 1.31 seconds)
## Iteration 1600: error is 0.350465 (50 iterations in 1.32 seconds)
## Iteration 1650: error is 0.349495 (50 iterations in 1.31 seconds)
## Iteration 1700: error is 0.349055 (50 iterations in 1.31 seconds)
## Iteration 1750: error is 0.348434 (50 iterations in 1.32 seconds)
## Iteration 1800: error is 0.347429 (50 iterations in 1.32 seconds)
## Iteration 1850: error is 0.346802 (50 iterations in 1.33 seconds)
## Iteration 1900: error is 0.345972 (50 iterations in 1.32 seconds)
## Iteration 1950: error is 0.345521 (50 iterations in 1.32 seconds)
## Iteration 2000: error is 0.345323 (50 iterations in 1.32 seconds)
## Iteration 2050: error is 0.344096 (50 iterations in 1.31 seconds)
## Iteration 2100: error is 0.343662 (50 iterations in 1.30 seconds)
## Iteration 2150: error is 0.343477 (50 iterations in 1.30 seconds)
## Iteration 2200: error is 0.342552 (50 iterations in 1.30 seconds)
## Iteration 2250: error is 0.342320 (50 iterations in 1.29 seconds)
## Iteration 2300: error is 0.342100 (50 iterations in 1.31 seconds)
## Iteration 2350: error is 0.341732 (50 iterations in 1.29 seconds)
## Iteration 2400: error is 0.341161 (50 iterations in 1.29 seconds)
## Iteration 2450: error is 0.340718 (50 iterations in 1.29 seconds)
## Iteration 2500: error is 0.340353 (50 iterations in 1.28 seconds)
## Iteration 2550: error is 0.339976 (50 iterations in 1.28 seconds)
## Iteration 2600: error is 0.339730 (50 iterations in 1.28 seconds)
## Iteration 2650: error is 0.338952 (50 iterations in 1.28 seconds)
## Iteration 2700: error is 0.338319 (50 iterations in 1.28 seconds)
## Iteration 2750: error is 0.338474 (50 iterations in 1.29 seconds)
## Iteration 2800: error is 0.338536 (50 iterations in 1.28 seconds)
## Iteration 2850: error is 0.337816 (50 iterations in 1.29 seconds)
## Iteration 2900: error is 0.337793 (50 iterations in 1.28 seconds)
## Iteration 2950: error is 0.337310 (50 iterations in 1.28 seconds)
## Iteration 3000: error is 0.336322 (50 iterations in 1.27 seconds)
## Iteration 3050: error is 0.336193 (50 iterations in 1.27 seconds)
## Iteration 3100: error is 0.335507 (50 iterations in 1.27 seconds)
## Iteration 3150: error is 0.335177 (50 iterations in 1.27 seconds)
## Iteration 3200: error is 0.334965 (50 iterations in 1.26 seconds)
## Iteration 3250: error is 0.334707 (50 iterations in 1.26 seconds)
## Iteration 3300: error is 0.334221 (50 iterations in 1.26 seconds)
## Iteration 3350: error is 0.333693 (50 iterations in 1.26 seconds)
## Iteration 3400: error is 0.333357 (50 iterations in 1.26 seconds)
## Iteration 3450: error is 0.332917 (50 iterations in 1.26 seconds)
## Iteration 3500: error is 0.332528 (50 iterations in 1.27 seconds)
## Iteration 3550: error is 0.332240 (50 iterations in 1.26 seconds)
```

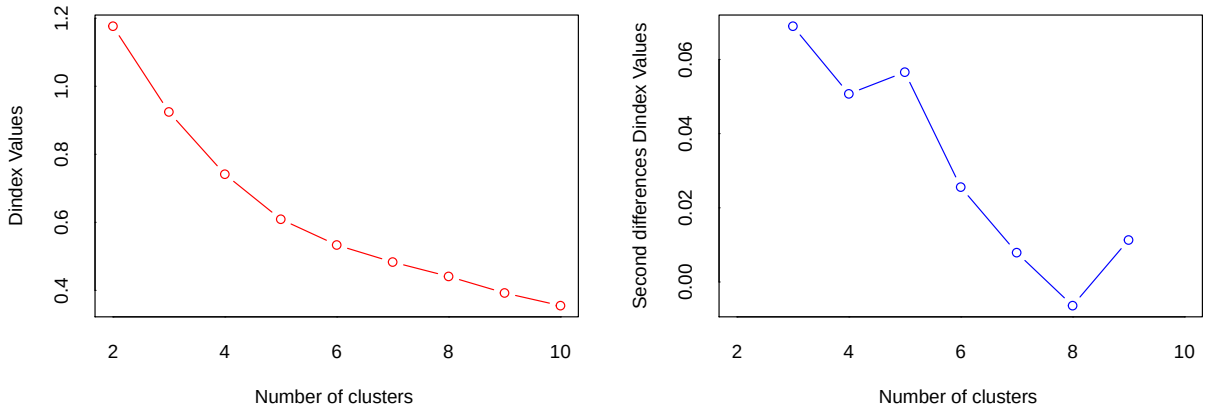
```
## Iteration 3600: error is 0.331911 (50 iterations in 1.26 seconds)
## Iteration 3650: error is 0.331429 (50 iterations in 1.26 seconds)
## Iteration 3700: error is 0.331002 (50 iterations in 1.26 seconds)
## Iteration 3750: error is 0.331290 (50 iterations in 1.26 seconds)
## Iteration 3800: error is 0.331103 (50 iterations in 1.27 seconds)
## Iteration 3850: error is 0.331059 (50 iterations in 1.26 seconds)
## Iteration 3900: error is 0.330474 (50 iterations in 1.25 seconds)
## Iteration 3950: error is 0.330365 (50 iterations in 1.26 seconds)
## Iteration 4000: error is 0.330259 (50 iterations in 1.26 seconds)
## Iteration 4050: error is 0.329687 (50 iterations in 1.26 seconds)
## Iteration 4100: error is 0.329197 (50 iterations in 1.25 seconds)
## Iteration 4150: error is 0.329115 (50 iterations in 1.25 seconds)
## Iteration 4200: error is 0.328642 (50 iterations in 1.25 seconds)
## Iteration 4250: error is 0.328504 (50 iterations in 1.25 seconds)
## Iteration 4300: error is 0.327821 (50 iterations in 1.26 seconds)
## Iteration 4350: error is 0.327790 (50 iterations in 1.25 seconds)
## Iteration 4400: error is 0.327426 (50 iterations in 1.25 seconds)
## Iteration 4450: error is 0.327153 (50 iterations in 1.24 seconds)
## Iteration 4500: error is 0.327016 (50 iterations in 1.24 seconds)
## Iteration 4550: error is 0.326814 (50 iterations in 1.25 seconds)
## Iteration 4600: error is 0.326841 (50 iterations in 1.23 seconds)
## Iteration 4650: error is 0.326133 (50 iterations in 1.24 seconds)
## Iteration 4700: error is 0.325394 (50 iterations in 1.23 seconds)
## Iteration 4750: error is 0.324997 (50 iterations in 1.24 seconds)
## Iteration 4800: error is 0.324301 (50 iterations in 1.23 seconds)
## Iteration 4850: error is 0.324243 (50 iterations in 1.24 seconds)
## Iteration 4900: error is 0.323709 (50 iterations in 1.24 seconds)
## Iteration 4950: error is 0.323329 (50 iterations in 1.24 seconds)
## Iteration 5000: error is 0.323220 (50 iterations in 1.23 seconds)
## Iteration 5050: error is 0.323047 (50 iterations in 1.24 seconds)
## Iteration 5100: error is 0.323210 (50 iterations in 1.23 seconds)
## Iteration 5150: error is 0.322484 (50 iterations in 1.24 seconds)
## Iteration 5200: error is 0.322516 (50 iterations in 2.49 seconds)
## Iteration 5250: error is 0.321826 (50 iterations in 1.22 seconds)
## Iteration 5300: error is 0.321676 (50 iterations in 1.23 seconds)
## Iteration 5350: error is 0.321161 (50 iterations in 1.22 seconds)
## Iteration 5400: error is 0.317922 (50 iterations in 1.21 seconds)
## Iteration 5450: error is 0.322531 (50 iterations in 1.21 seconds)
## Iteration 5500: error is 0.321680 (50 iterations in 1.21 seconds)
## Iteration 5550: error is 0.320561 (50 iterations in 1.19 seconds)
## Iteration 5600: error is 0.319832 (50 iterations in 1.21 seconds)
## Iteration 5650: error is 0.319539 (50 iterations in 1.21 seconds)
## Iteration 5700: error is 0.319560 (50 iterations in 1.23 seconds)
## Iteration 5750: error is 0.319469 (50 iterations in 1.22 seconds)
## Iteration 5800: error is 0.319163 (50 iterations in 1.23 seconds)
## Iteration 5850: error is 0.318570 (50 iterations in 1.23 seconds)
## Iteration 5900: error is 0.318707 (50 iterations in 1.22 seconds)
## Iteration 5950: error is 0.318405 (50 iterations in 1.22 seconds)
## Iteration 6000: error is 0.318129 (50 iterations in 1.21 seconds)
## Iteration 6050: error is 0.318200 (50 iterations in 1.21 seconds)
## Iteration 6100: error is 0.317680 (50 iterations in 1.21 seconds)
## Iteration 6150: error is 0.317567 (50 iterations in 1.21 seconds)
## Iteration 6200: error is 0.317071 (50 iterations in 1.20 seconds)
## Iteration 6250: error is 0.316929 (50 iterations in 1.20 seconds)
```

```
## Iteration 6300: error is 0.316830 (50 iterations in 1.20 seconds)
## Iteration 6350: error is 0.317080 (50 iterations in 1.20 seconds)
## Iteration 6400: error is 0.316980 (50 iterations in 1.19 seconds)
## Iteration 6450: error is 0.317358 (50 iterations in 1.20 seconds)
## Iteration 6500: error is 0.317509 (50 iterations in 1.19 seconds)
## Iteration 6550: error is 0.317676 (50 iterations in 1.21 seconds)
## Iteration 6600: error is 0.317375 (50 iterations in 1.22 seconds)
## Iteration 6650: error is 0.317161 (50 iterations in 1.25 seconds)
## Iteration 6700: error is 0.317615 (50 iterations in 1.22 seconds)
## Iteration 6750: error is 0.317859 (50 iterations in 1.23 seconds)
## Iteration 6800: error is 0.317742 (50 iterations in 1.23 seconds)
## Iteration 6850: error is 0.317788 (50 iterations in 1.23 seconds)
## Iteration 6900: error is 0.317464 (50 iterations in 1.22 seconds)
## Iteration 6950: error is 0.317730 (50 iterations in 1.22 seconds)
## Iteration 7000: error is 0.317459 (50 iterations in 1.22 seconds)
## Iteration 7050: error is 0.317730 (50 iterations in 1.21 seconds)
## Iteration 7100: error is 0.317829 (50 iterations in 1.22 seconds)
## Iteration 7150: error is 0.317998 (50 iterations in 1.22 seconds)
## Iteration 7200: error is 0.317934 (50 iterations in 1.22 seconds)
## Iteration 7250: error is 0.318055 (50 iterations in 1.21 seconds)
## Iteration 7300: error is 0.317903 (50 iterations in 1.22 seconds)
## Iteration 7350: error is 0.317824 (50 iterations in 1.22 seconds)
## Iteration 7400: error is 0.318157 (50 iterations in 1.22 seconds)
## Iteration 7450: error is 0.317982 (50 iterations in 1.22 seconds)
## Iteration 7500: error is 0.318329 (50 iterations in 1.23 seconds)
## Iteration 7550: error is 0.318297 (50 iterations in 1.22 seconds)
## Iteration 7600: error is 0.318516 (50 iterations in 1.23 seconds)
## Iteration 7650: error is 0.318308 (50 iterations in 1.22 seconds)
## Iteration 7700: error is 0.317973 (50 iterations in 1.23 seconds)
## Iteration 7750: error is 0.318368 (50 iterations in 1.24 seconds)
## Iteration 7800: error is 0.318510 (50 iterations in 1.25 seconds)
## Iteration 7850: error is 0.318316 (50 iterations in 1.24 seconds)
## Iteration 7900: error is 0.318406 (50 iterations in 1.24 seconds)
## Iteration 7950: error is 0.318532 (50 iterations in 1.24 seconds)
## Iteration 8000: error is 0.318292 (50 iterations in 1.24 seconds)
## Iteration 8050: error is 0.318299 (50 iterations in 1.24 seconds)
## Iteration 8100: error is 0.318081 (50 iterations in 1.23 seconds)
## Iteration 8150: error is 0.317847 (50 iterations in 1.24 seconds)
## Iteration 8200: error is 0.317786 (50 iterations in 1.25 seconds)
## Iteration 8250: error is 0.317737 (50 iterations in 1.24 seconds)
## Iteration 8300: error is 0.317886 (50 iterations in 1.24 seconds)
## Iteration 8350: error is 0.317863 (50 iterations in 1.24 seconds)
## Iteration 8400: error is 0.318152 (50 iterations in 1.24 seconds)
## Iteration 8450: error is 0.318140 (50 iterations in 1.24 seconds)
## Iteration 8500: error is 0.318260 (50 iterations in 1.24 seconds)
## Iteration 8550: error is 0.318229 (50 iterations in 1.25 seconds)
## Iteration 8600: error is 0.318105 (50 iterations in 1.26 seconds)
## Iteration 8650: error is 0.317957 (50 iterations in 1.24 seconds)
## Iteration 8700: error is 0.317979 (50 iterations in 1.26 seconds)
## Iteration 8750: error is 0.317578 (50 iterations in 1.25 seconds)
## Iteration 8800: error is 0.317431 (50 iterations in 1.26 seconds)
## Iteration 8850: error is 0.317585 (50 iterations in 1.25 seconds)
## Iteration 8900: error is 0.317498 (50 iterations in 1.26 seconds)
## Iteration 8950: error is 0.317322 (50 iterations in 1.26 seconds)
```

```
## Iteration 9000: error is 0.317551 (50 iterations in 1.26 seconds)
## Iteration 9050: error is 0.317543 (50 iterations in 1.24 seconds)
## Iteration 9100: error is 0.317395 (50 iterations in 1.26 seconds)
## Iteration 9150: error is 0.317755 (50 iterations in 1.25 seconds)
## Iteration 9200: error is 0.317711 (50 iterations in 1.26 seconds)
## Iteration 9250: error is 0.317970 (50 iterations in 1.24 seconds)
## Iteration 9300: error is 0.317600 (50 iterations in 1.25 seconds)
## Iteration 9350: error is 0.317182 (50 iterations in 1.25 seconds)
## Iteration 9400: error is 0.317321 (50 iterations in 1.26 seconds)
## Iteration 9450: error is 0.317280 (50 iterations in 1.26 seconds)
## Iteration 9500: error is 0.317273 (50 iterations in 1.27 seconds)
## Iteration 9550: error is 0.317200 (50 iterations in 1.26 seconds)
## Iteration 9600: error is 0.317154 (50 iterations in 1.26 seconds)
## Iteration 9650: error is 0.316753 (50 iterations in 1.26 seconds)
## Iteration 9700: error is 0.317313 (50 iterations in 1.25 seconds)
## Iteration 9750: error is 0.317194 (50 iterations in 1.26 seconds)
## Iteration 9800: error is 0.317244 (50 iterations in 1.27 seconds)
## Iteration 9850: error is 0.317173 (50 iterations in 1.26 seconds)
## Iteration 9900: error is 0.317133 (50 iterations in 1.27 seconds)
## Iteration 9950: error is 0.316665 (50 iterations in 1.26 seconds)
## Iteration 10000: error is 0.316441 (50 iterations in 1.26 seconds)
## Fitting performed in 255.39 seconds.
```

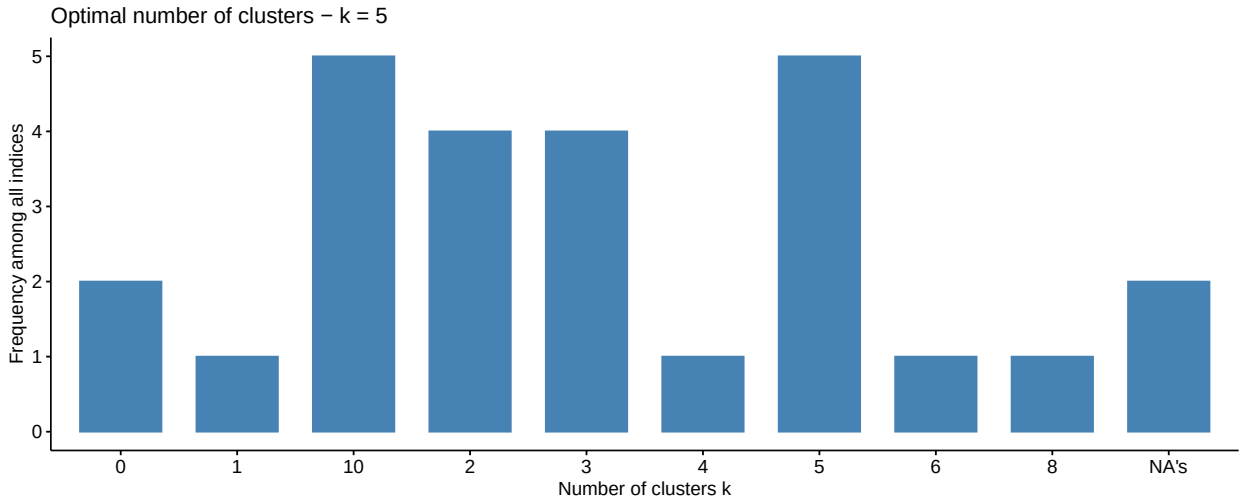



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                In the plot of Hubert index, we seek a significant knee that corresponds to a
##                significant increase of the value of the measure i.e the significant peak in Hubert
##                index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 4 proposed 2 as the best number of clusters
## * 4 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 5 proposed 5 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 5 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  5
##
##
## *******************************************************************

## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 4 proposed  2 as the best number of clusters
## * 4 proposed  3 as the best number of clusters
## * 1 proposed  4 as the best number of clusters
## * 5 proposed  5 as the best number of clusters
## * 1 proposed  6 as the best number of clusters
## * 1 proposed  8 as the best number of clusters
## * 5 proposed  10 as the best number of clusters
## * 2 proposed  NA's as the best number of clusters
##
## Conclusion
## ==========================
## * According to the majority rule, the best number of clusters is  5 .
```



2.3.1 **Validación**

Se busca cuantificar la homogeneidad dentro de cada cluster y a su vez la separación entre los demás, teniendo en cuenta que ambos criterios tienen tendencias opuestas, es decir, a mayor número de clusters, mayor homogeneidad pero menor distancia, es una forma de saber que tan bueno es el resultado. Para ello los dos índices mayormente utilizados son silhouette Width y Dunn pero también veremos las medidas de estabilidad.

Definiciones de homogeneidad de cluster:

- Promedio de la distancia entre todos los pares de observaciones:

  $Homogeneidad(C) = \sum Oi, Oj \in C, Oi \neq Oj \ distancia(Oi, Oj) \, \|C\| * (\|C\| - 1)$

- Promedio de la distancia entre las observaciones que forman el cluster y su centroide:

  $Homogeneidad(C) = \sum Oi \in C, \ distancia(Oi, O^-) \ \|C\|$

El Índice Silhouette:

Cuantifica la calidad de la asignación que se ha realizado de una observación comparando su semejanza a las demás observaciones del mismo cluster frente a las de los otros cluster. Su valor puede estar entre 1 y -1, siendo los valores altos un buen indicativo que la observación se ha asignado al cluster correcto, mientras los valores estén cercanos a cero, significa un valor medio entre dos clusters de la observación y por último si los valores son negativos quiere decir que se realizo una asignación incorrecta de la observación.

Para cada observación i, el $silhouette \ coeficient$ (s_i) se obtiene del siguiente modo:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

a_i media de las distancias entre la observación i y el resto de observaciones.

b_i es la menor de las distancias promedio entre i y el resto de clusters.

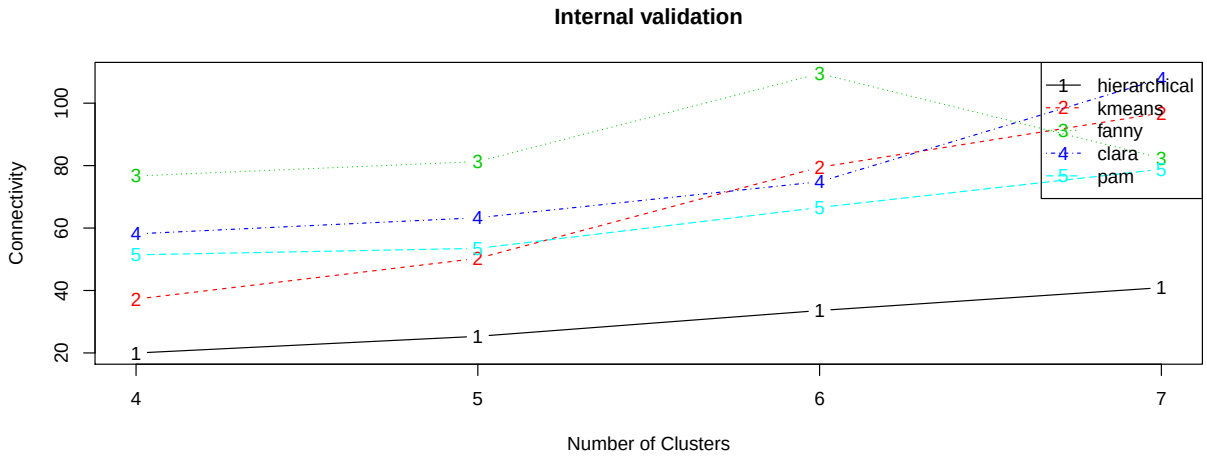
El índice Dunn:

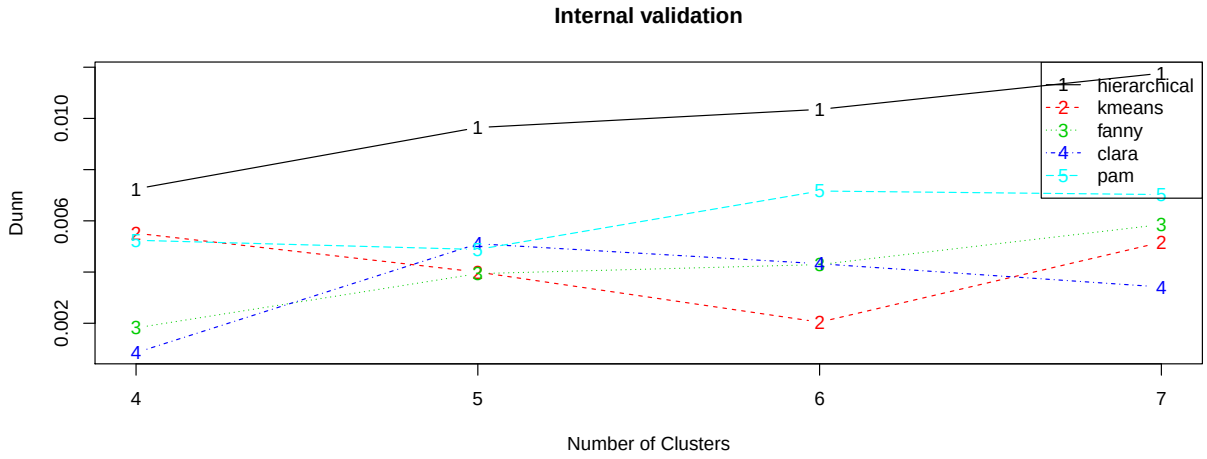
El objetivo con este índice es maximizar asignando valores grandes al númerador y pequeños al denominador, esto se logra si se tiene clusters compactos y bien separados; Sin embargo, este índice se sugiere ser utilizado "en el peor de los casos" puesto que su gran inconveniente es si alguno de los clusters no tiene un comportamiento ideal y su calidad es baja, cómo el denominador utiliza el máximo en un lugar de la media, el índice se vería totalmente influenciado por este ocultando a los demás.

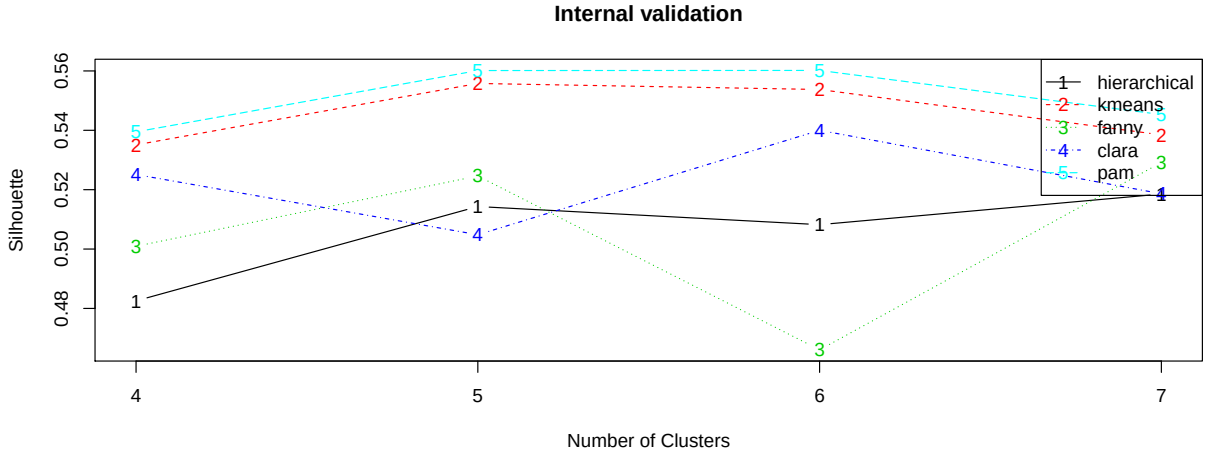
Calcular el índice Dunn como:

$$D = \frac{separacion \ minima \ interclusters}{separacion \ maxima \ intracluster}$$

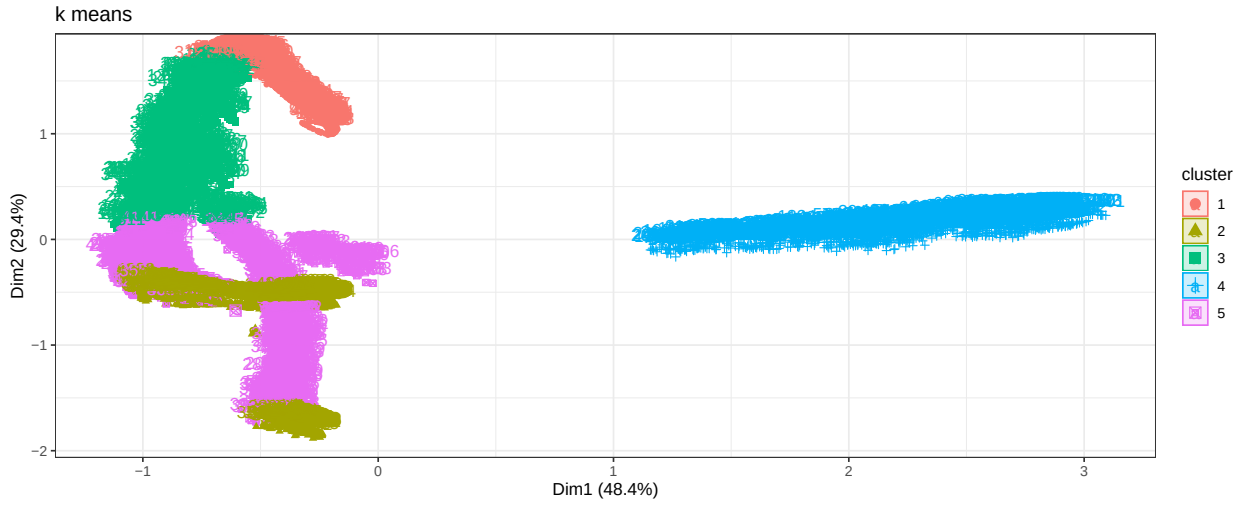
```
##
## Clustering Methods:
##  hierarchical kmeans fanny clara pam
##
## Cluster sizes:
##  4 5 6 7
##
## Validation Measures:
##                                  4        5        6        7
##
## hierarchical Connectivity   19.9758  25.3234  33.5837  40.9095
##              Dunn            0.0072   0.0096   0.0104   0.0118
##              Silhouette      0.4825   0.5143   0.5082   0.5185
## kmeans       Connectivity   37.2456  50.2913  79.4671  96.8440
##              Dunn            0.0055   0.0040   0.0020   0.0052
##              Silhouette      0.5351   0.5558   0.5537   0.5385
## fanny        Connectivity   76.6925  81.2861 109.4802  82.3925
##              Dunn            0.0018   0.0039   0.0043   0.0059
##              Silhouette      0.5009   0.5246   0.4660   0.5292
## clara        Connectivity   58.1480  63.3016  74.8365 107.8226
##              Dunn            0.0009   0.0051   0.0043   0.0034
##              Silhouette      0.5251   0.5048   0.5400   0.5186
## pam          Connectivity   51.4306  53.4861  66.5857  78.8393
##              Dunn            0.0052   0.0049   0.0072   0.0070
##              Silhouette      0.5395   0.5601   0.5602   0.5453
##
## Optimal Scores:
##
##              Score   Method       Clusters
## Connectivity 19.9758 hierarchical 4
## Dunn          0.0118 hierarchical 7
## Silhouette    0.5602 pam          6
```



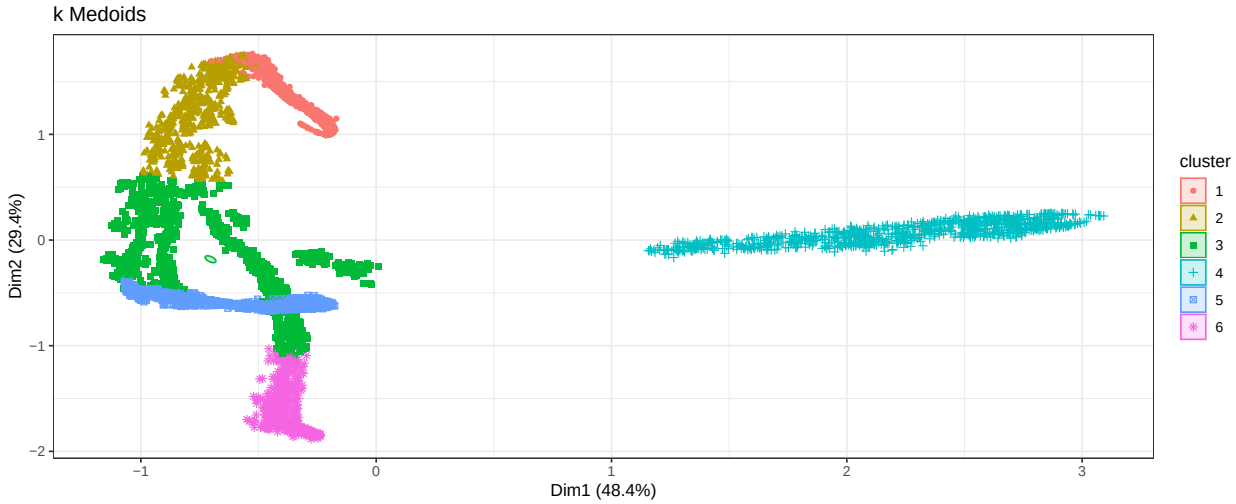




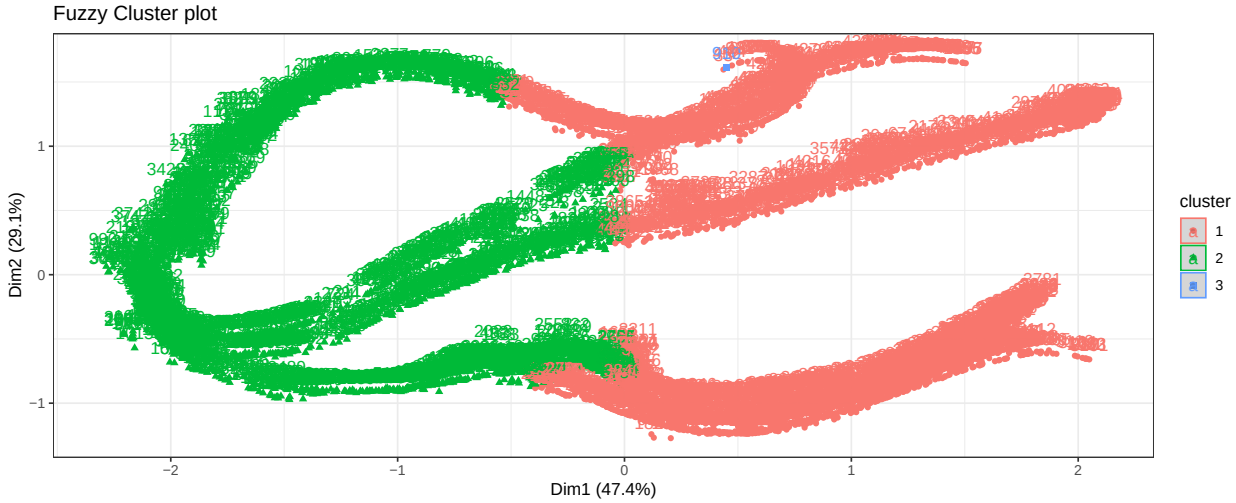
2.3.2 Gráfico **K means clustering**



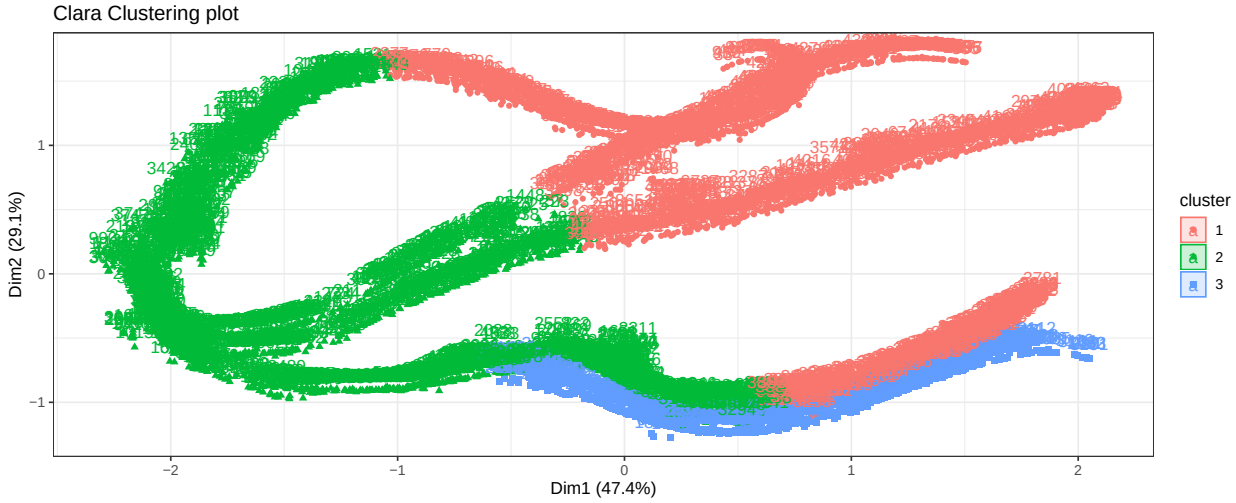
2.3.3 Gráfico **K medoids**



2.3.4 Gráfico **Fuzzy c-means clustering**



2.3.5 Gráfico **Clara Clustering**



4. Conclusiones

Una vez realizado la clusterización con los diferentes métodos y aplicando los test respectivos para la validación y selección de los mejores método, se obtiene:

- El mejor método para este grupo de datos es el jerarquico, seguido del K-Medoids, según los métodos evaluados.
- Cuándo se realiza el embebimiento, los cluster mejoran en su rendimiento y clasificación, lo que permite una mejor visualización y segmentación de cada uno de items en la base de datos.
- Se selecciona el método K-medoids ya que es quién gana en Silhouette y lo importante es encontrar una herramienta que pueda generalizar.
- Se decidió usar software estadístico `R 3.5.2` bajo la consola `RStudio 1.1.463` para los datos, ya que se encontró y fue posible implementar cada método con las librerías que tiene; permitiendo una visualización y compilamiento adecuada en cada uno.

5. Referencias

- [1] Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. Biometrics, 21, 768-769.
- [2] Arthur, David, and Sergi Vassilvitskii. "K-means++: The Advantages of Careful Seeding." SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. 2007, pp. 1027-1035.
- [3] Reynolds, A., Richards, G., de la Iglesia, B. and Rayward-Smith, V. (1992) Clustering rules: A comparison of partitioning and hierarchical clustering algorithms; Journal of Mathematical Modelling and Algorithms 5, 475-504. doi: 10.1007/s10852-005-9022-1.
- [4] The particular method fanny stems from chapter 4 of Kaufman and Rousseeuw (1990) (see the references in daisy) and has been extended by Martin Maechler to allow user specified memb.exp, iniMem.p, maxit, tol, etc.
- [5] R. Yager and D. Filev, "Generation of fuzzy rules by mountain clustering," J. of Intelligent and Fuzzy Systems, vol. 2, no. 3, pp. 209 - 219 (1994).
- [5] S. Chiu, "Method and software for extracting fuzzy classification rules by subtractive clustering", Fuzzy Information Processing Society, NAFIPS, pp. 461 - 465 (1996).
- [6] Kaufman and Rousseeuw (see agnes), originally. Metric "jaccard": Kamil Kozlowski (@ownedoutcomes.com) and Kamil Jadeszko. All arguments from trace on, and most R documentation and all tests by Martin Maechler.
- [7] Murtagh, Fionn and Legendre, Pierre (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion? Journal of Classification, 31, 274-295. doi: 10.1007/s00357-014-9161-z.
- [8] Maaten, L. Van Der, 2014. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research, 15, p.3221-3245.