Clustering (Versión incompleta)

Rodrigo Meneses Orellana

9 de octubre de 2019

Introducción

El clustering es un **conjunto de técnicas de minería de datos** provenientes de la **ciencia de datos** y corresponden a **aprendizaje estadístico no supervisado**, su principal fundamento son las **reglas de asociación** que permiten identificar regiones en la información con altas densidades de datos, como puede apreciarse a continuación.

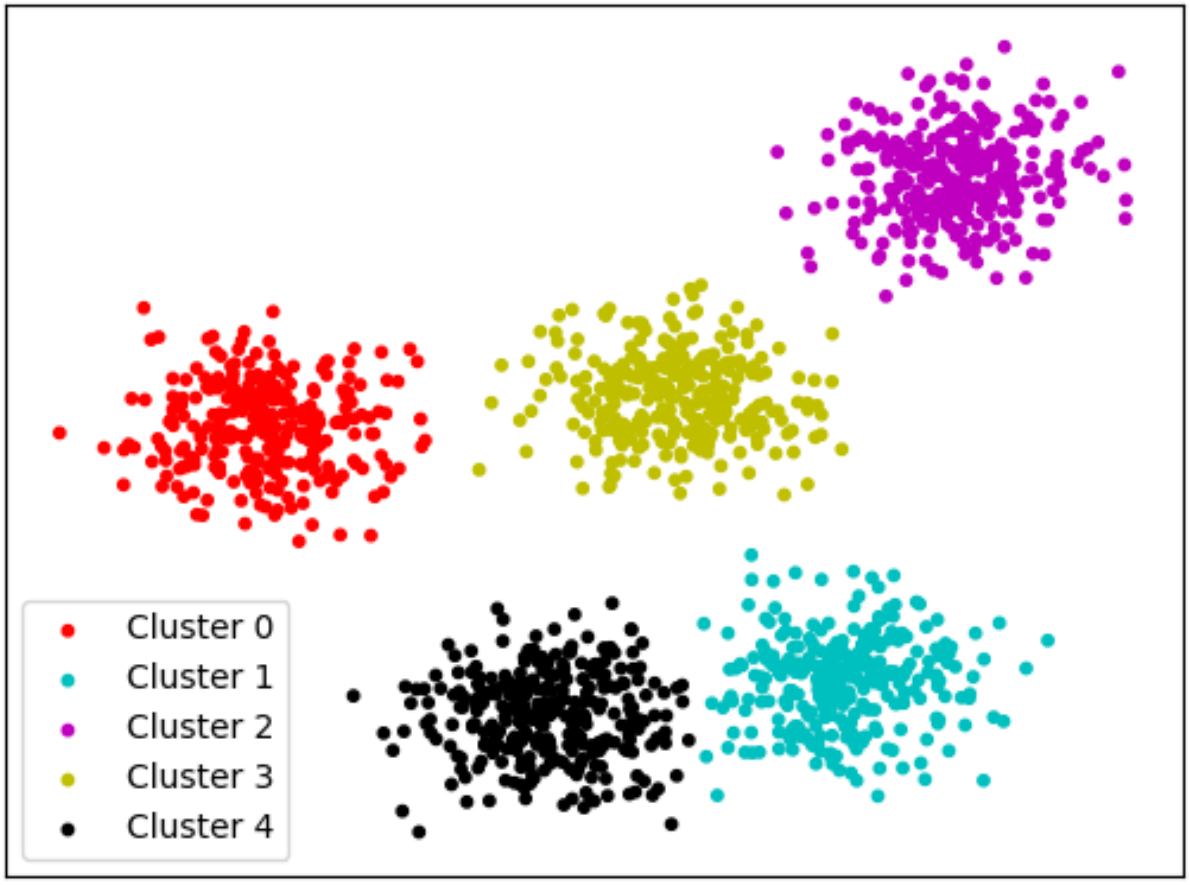

Figura 1: Ejemplo de clusters

Estas regiones tienen tantas dimensiones como variables tenga la información.

En el presente texto se revisarán las tres ramas principales del clustering:

- Partitioning Clustering: rama donde el usuario conoce o define el número de clusters por generar y en la cual se encuentran algoritmos como *k-means, k-medoids* y *CLARA*.

- Hierarchical Clustering: rama contraria al Partitioning Clustering donde el usuario no necesita especificar el número de clusters y en la cual se encuentran algoritmos como el *agglomerative clustering* y el *divisive clustering*.
- Mixture models: rama construida por la combinación de las ramas anteriores y en la cual se encuentran algoritmos como *model based clustering*, *fuzzy clustering* y *hierarchical k-means*.

Aunque dado que existe una amplia investigación en el área de la ciencia de datos, es posible que ya existan más algoritmos.

Recursos

En **R** se cuentan con múltiples librerías que implementan elementos del clustering.

- **stats**: la librería estadística base de R debida al **Core Team** de R.
- **cluster**: debida a **Martin Maechler** miembro del **R Core Development Team** y coautores como **Anja Struyf** y **Mia Hubert**.
- **mclust**: debida a **Chris Fraley, Adrian E. Raftery** y **Luca Scrucca**.
- **factoextra**: debida a **Alboukadel Kassambara** y **Fabian Mundt**.
- **dendextend**: debida a **Tal Galili**.

Se hará uso de todas ellas.

Reglas de asociación

Para cada algoritmo de clustering es necesario definir la(s) regla(s) de asociación, estas permiten llevar a cabo las agrupaciones de la información en los *clusters*. En general serán empleadas como reglas de asociación las medidas de distancia, que permitirán medir la similitud o diferencia entre observaciones. La selección de dicha medida es muy importante, pues debe ser adecuada para los datos y la situación con la que se trabaja.

Se enlistan algunas medidas de distancia frecuentes:

- **Distancia euclidiana generalizada**: sean x, y elementos de un espacio n-dimensional, entonces la distancia euclidiana generalizada se define como $d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$.
- **Distancia l_1 generalizada**: sean x, y elementos de un espacio n-dimensional, entonces la distancia l_1 generalizada se define como $d(x,y) = \sum |x_i - y_i|$.
- **Distancia de correlación lineal**: sean x, y elementos de un espacio n-dimensional, entonces la distancia de correlación lineal se define como $d(x,y) = 1 - cor(x,y)$ donde la correlación puede ser a través de cualquier método de correlación lineal como Pearson, Spearman, Kendall, etcétera.
- **Distancia de correlación Jackknife**: similar a la anterior pero permite atenuar el efecto global en la correlación lineal debido a valores extremos (*outliers*) cuando estos no son demasiados. Sean x, y elementos de un espacio n-dimensional, se obtiene la correlación lineal dado que se ha eliminado la i-ésima observación obteniendo r_i, luego se repite dicho proceso para cada i en $[1,n]$, finalmente se saca el promedio de dichos r_i: $r(x,y) = \frac{1}{n} \sum_{i=1}^{n}$ y luego la distancia de correlación Jackknife se define como $d(x,y) = 1 - r(x,y)$ donde la correlación r_i puede ser a través de cualquier método de correlación lineal como Pearson, Spearman, Kendall, etcétera.
- **Distancia de emparejamiento simple**: sean dos individuos A, B con n características binarias diferentes (género, por ejemplo), se define el coeficiente de emparejamiento simple como $SMC(A,B) = \frac{M}{C}$, donde M="características que comparten" y C="Total de características". Luego, se define la distancia de emparejamiento simple de A y B como $d(A,B) = 1 - SMC(A,B)$.

- **Distancia de Jaccard**: similar al anterior aunque, condicionado a alguno de los dos atributos de cada característica, es decir, si una característica es género y condiciono la coincidencia a "hombre", entonces, para un par de sujetos A, B, la coincidencia se cuenta si ambos son hombres, pero si ambos son mujeres no se cuenta, así como tampoco si uno es hombre y otro mujer. Por lo tanto, se define el índice de Jaccard como $J(A,B) = \frac{M}{C}$, donde M="características que comparten sujetas a una dominancia binaria" y C="Total de características". Luego, se define la distancia de Jaccard de A y B como $d(A,B) = 1 - J(A,B)$.
- **Distancia coseno**: el coseno es una medida adecuada para comprender la similitud entre las direcciones de dos vectores n dimensionales. Sean x, y elementos de un espacio n-dimensional, entonces se define la distancia coseno como $d(x,y) = cos(\theta)$ donde θ es el ángulo que formal los vectores.

En general, será util reescalar las observaciones con la finalidad de que su escala de medición no afecte las observaciones (siempre que estas observaciones sean continuas, no tiene sentido reescalar factores).

Calculando distancias

Se hará uso del conjunto de datos `iris` para ejemplificar algunas de las distancias anteriores, antes de comenzar habra un script nuevo y coloque el siguiente header para tener todos los recursos necesarios disponibles y activados.

```
list.of.packages <- c("ggplot2","cluster","mclust","factoextra",
                      "dendextend","ggpubr","tidyverse")
new.packages<-list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

require(ggplot2)
require(cluster)
require(mclust)
require(factoextra)
require(dendextend)
require(ggpubr)
require(tidyverse)
```

Se da un vistazo a las variables y datos de `iris`.

```
kable(head(iris), format = "latex",
      caption = "Muestra de la información de iris",
      booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = "HOLD_position")
```

Tabla 1: Muestra de la información de iris

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Será añadido además, un factor del color de las plantas (información ficticia) con el fin de dar más elementos para los cálculos posteriores, así como un identificador por planta como nombre de fila.

```
db<-iris
colores<-colors()[c(1,2)]
color<-c()
set.seed(123)
for(i in 1:nrow(db)){color[i]<-colores[round(runif(1,1,2),0)]}
db$Color<-as.factor(color)

# Nombres de fila

id<-paste0("Planta",1:nrow(db))
row.names(db)<-id
```

Se rescalan los datos y se calculan a continuación algunas de las distancias.

```
# Reescalamiento para evitar sesgo de escala

db[,1:4]<-scale(db[,1:4])

# Distancia euclideana generalizada (solo tiene sentido para los valores numéricos)

euc_dist_original<-dist(db[,1:4])
euc_dist<-as.matrix(euc_dist_original)[1:5,1:5] # Seleccionamos solo una muestra
euc_dist<-round(euc_dist,2)

# Distancia de correlación lineal

cor_dist_original<-get_dist(db[,1:4],method = "pearson")
cor_dist<-round(as.matrix(cor_dist_original),2)
cor_dist<-cor_dist[1:5,1:5]

# Distancia coseno

coseno <- function(x, y){
  resultado <- x%*%y / (sqrt(x %*% x) * sqrt(y %*%y ))
  return(as.numeric(resultado))
}

distancias<-list()
for (i in 1:nrow(db)) {
      distancias[[i]]<-1
  for(j in 1:nrow(db)){
distancias[[i]][j]<-1
      distancias[[i]][j]<-1-coseno(as.vector(t(db[i,1:4])),as.vector(t(db[j,1:4])))
  }
}
```

Sin embargo, las distancias de Jaccard o emparejamiento simple no son posibles de obtener por la naturaleza de los datos.

Partitioning Clustering

Clustering basado en que previamente el usuario conoce el número de clusters por generar.

El algoritmo k-means

El algoritmo **k-means** tiene como finalidad que dada K (el número de clusters que define el usuario), se asigne cada una de las observaciones a alguno de los K clusters de tal manera que la varianza interna de cada cluster se minimice, esta varianza está dada por las reglas de asociación que se definen entre las observaciones. En el caso del algoritmo k-means, la regla de asociación usual es la **distancia euclidiana generalizada** y entonces, para todas las observaciones que se hayan asignado por cada cluster, será calculada dicha distancia respecto al centroide del grupo y se buscará minimizar el promedio (la varianza intra-clusters). En palabras simples, el centroide respecto al cual se calculan las distancias es un punto equidistante a todas las observaciones que están contenidas en el cluster, lo cual se expresa como el punto cuyas coordenadas en el espacio están dadas por la media de cada variable del conjunto de datos en el cluster.

El algoritmo puede ser resumido en los siguientes pasos.

1. Aleatoriamente asignar cada observación a alguno de los K clusters definidos (solución inicial).
2. Calcular el centroide de cada cluster.
3. Obtener las distancias de cada observación respecto al centroide de cada cluster.
4. Reasignar la observación al cluster donde la distancia de la observación al centroide del cluster sea la mínima.
5. Obtener el promedio de las distancias por cluster.

Los pasos 2-5 se repiten de manera iterativa hasta que el promedio de las distancias por cluster (5) no se reduzca más de manera significativa o hasta que se alcance un límite de iteraciones dadas por el usuario.

Sin embargo, cuán óptima sea la solución depende en gran medida de la solución inicial dada, por lo tanto, debe repertirse el algoritmo múltiples veces de forma completa para hallar una solución lo más independiente de la selección inicial, es por ello que el algoritmo k-means es doblemente iterativo.

Ventajas

- Rapidez y sencillez al aplicarlo en conjuntos de datos numéricos.
- Resultados suficientemente fiables cuando no hay demasiados valores extremos (*outliers*)

Desventajas

- La necesidad de asignar el número de clusters.
- La dependencia de la solución final respecto a la solución inicial.
- El modelo es reproducible únicamente a través de semillas por lo que documentarlo puede ser difícil.
- Es poco robusto frente a altas cantidades de valores extremos (*outliers*).

Ejemplo

A continuación se muestra un ejemplo de la aplicación de k-means en `iris` usando la información sobre el sépalo, es decir, `Sepal.Length` y `Sepal.Width`, recuerde que los datos ya se encuentran estandarizados. Se tomará como agrupamiento de contraste a la especie (variable **Species**), únicamente para comprender si este factor define clusters adecuados, lo cual debería ser así, pues se tratan de la misma especie.

```
datos<-db[,c(1,2,5)] # Datos continuos de iris (solo dos variables)
rownames(datos)<-NULL
set.seed(333) # Semilla para asegurar la reproducibilidad del resultado

# K-means
```

```r
agrupamiento<-kmeans(x=datos[,1:2],centers = 3,nstart = 50)
agrupamiento
```

```
## K-means clustering with 3 clusters of sizes 56, 49, 45
##
## Cluster means:
##   Sepal.Length Sepal.Width
## 1  -0.05880023  -0.9017619
## 2  -0.99872072   0.9032290
## 3   1.16066951   0.1386766
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 3 3 3 1 3 1 3 1 3 1 1 1 1 1 1 3 1 1 1 1
##  [71] 3 1 1 1 3 3 3 3 1 1 1 1 1 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3
## [106] 3 1 3 1 3 3 1 3 1 1 3 3 3 3 1 3 1 3 1 3 3 1 1 1 3 3 3 1 1 1 3 3 3 1 3
## [141] 3 3 1 3 3 3 1 3 3 1
##
## Within cluster sum of squares by cluster:
## [1] 34.32476 38.72457 28.88194
##  (between_SS / total_SS =  65.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

El objeto producto de la función `kmeans()` genera:

- El número de clusters y su tamaño (cantidad de observaciones asignadas por cluster)
- Las medias por cluster por variable (2 variables y 3 clusters equivalen a 6 medias)
- El vector de asignación donde se detalla a qué cluster ha ido asignado cada observación
- El vector de sumas de distancias intra-clusters que corresponde a lo mencionado en el paso 5 del algoritmo pero, reemplazando el promedio por la simple suma de distancias (cuando se desee penalizar a los clusters con pocas observaciones deberá preferirse el promedio) donde también se incluye la varianza entre grupos, la cual habla sobre la calidad del modelo y es equivalente al coeficiente de determinación R^2, sin embargo, esta medida es sesgada por el número de clusters por lo cual no debería ser tomada en cuenta de manera definitiva para hablar sobre la calidad del algoritmo.

Se muestra gráficamente el agrupamiento hecho por `kmeans` donde el color es el agrupamiento de contraste (especie).

```r
datos <- datos %>% mutate(cluster = agrupamiento$cluster)
datos <- datos %>% mutate(cluster = as.factor(cluster),
                          grupo   = as.factor(Species))

ggplot(data = datos, aes(x = datos$Sepal.Length, y = datos$Sepal.Width, color = grupo)) +
  geom_text(aes(label = cluster), size = 5) +
  theme_bw() +
  theme(legend.position = "none")
```

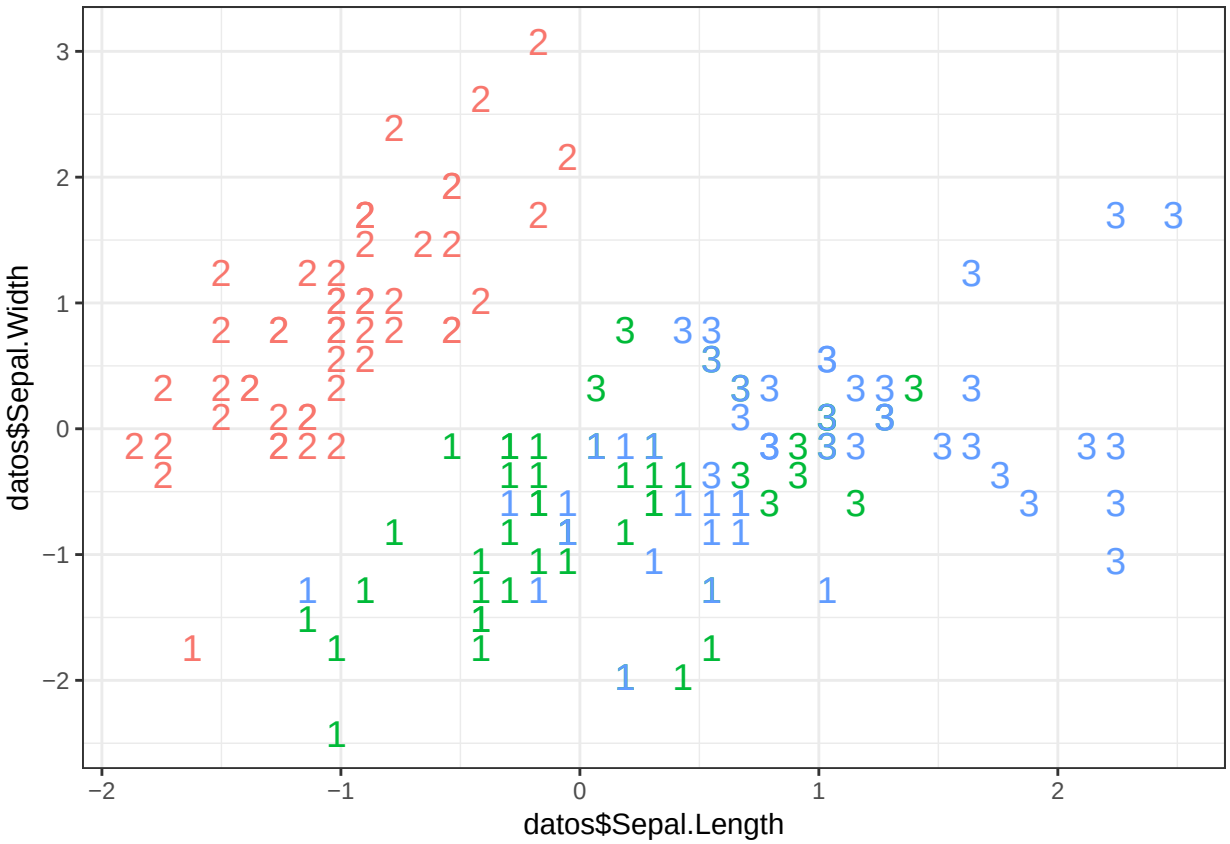

Figura 2: Clusters de iris (sépalo) con agrupamiento de contraste

Se ve una amplia confusión del algoritmo entre los clusters 1 y 3 respecto al agrupamiento de contraste, lo cual puede ser validado mediante una matriz de confusión.

```
print.table(table(agrupamiento$cluster,datos[,"Species"],dnn=list("Clusters","Especies")))
```

```
##          Especies
## Clusters setosa versicolor virginica
##        1      1         36        19
##        2     49          0         0
##        3      0         14        31
```

Esto quiere decir que el agrupamiento de contraste no genera agrupaciones correctas, mientras que los clusters que genera el algoritmo lucen bien. Esto genera una conclusión importante, la cual es que **el tamaño del sépalo no está dominado por la especie** y por lo tanto, agrupar por especie esperando tamaños similares de sépalo es un supuesto no general (válido en el caso del cluster 2, pero no en los otros dos).

En el siguiente gráfico se observan los clusters sin considerar el agrupamiento de contraste.

```
ggplot(data = datos, aes(x = datos$Sepal.Length, y = datos$Sepal.Width, color = cluster))+
  geom_text(aes(label = cluster), size = 5) +
```

```
  theme_bw() +
  theme(legend.position = "none")
```

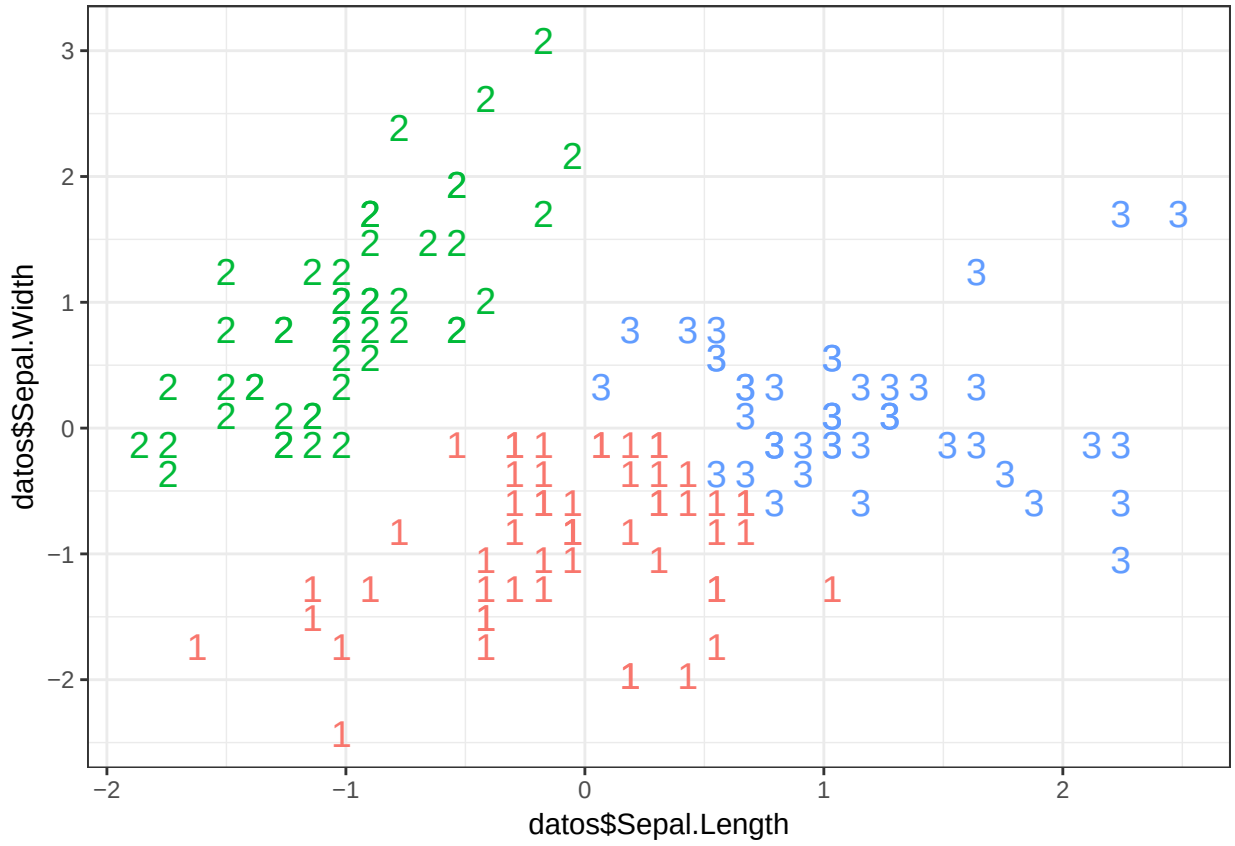

Figura 3: Clusters de iris (sépalo)

Puede apreciarse como la presencia de valores extremos (pese a que las variables están estandarizadas) podría ser un problema y como sin su presencia el cluster 1 y 3 podrían ser uno solo.

En altas dimensiones (más de 3) no será posible obtener una representación gráfica completamente fiable de los clusters (pueden aplicarse métodos de reducción de dimensiones pero, en general no es muy necesario), en ese caso **inferir el número de clusters** es una tarea más complicada, pues no se podrá basar la selección en la simple observación del conjunto de datos.

En dicho caso, se usa la misma lógica del algoritmo, realizar el algoritmo para cierto número de iteraciones y para cierto conjunto de valores de K, posteriormente identificar el valor de K para el cual la varianza intra-clusters ya no se reduce significativamente a partir de el; es este valor de K el que podría considerarse óptimo. Este proceso en términos teóricos es increíblemente complicado, sin embargo, computacionalmente es sencillo.

A partir de la función `fviz_nbclust` de la librería `factoextra`, se hará uso de los datos del pétalo y sépalo en `iris` para ejemplificar el uso de la función.

```
datos<-db[,1:4]
k_optima<-fviz_nbclust(datos,FUNcluster = kmeans,method = "wss",k.max=20,
          diss = get_dist(datos,method = "euclidian"),nstart=50)
k_optima
```

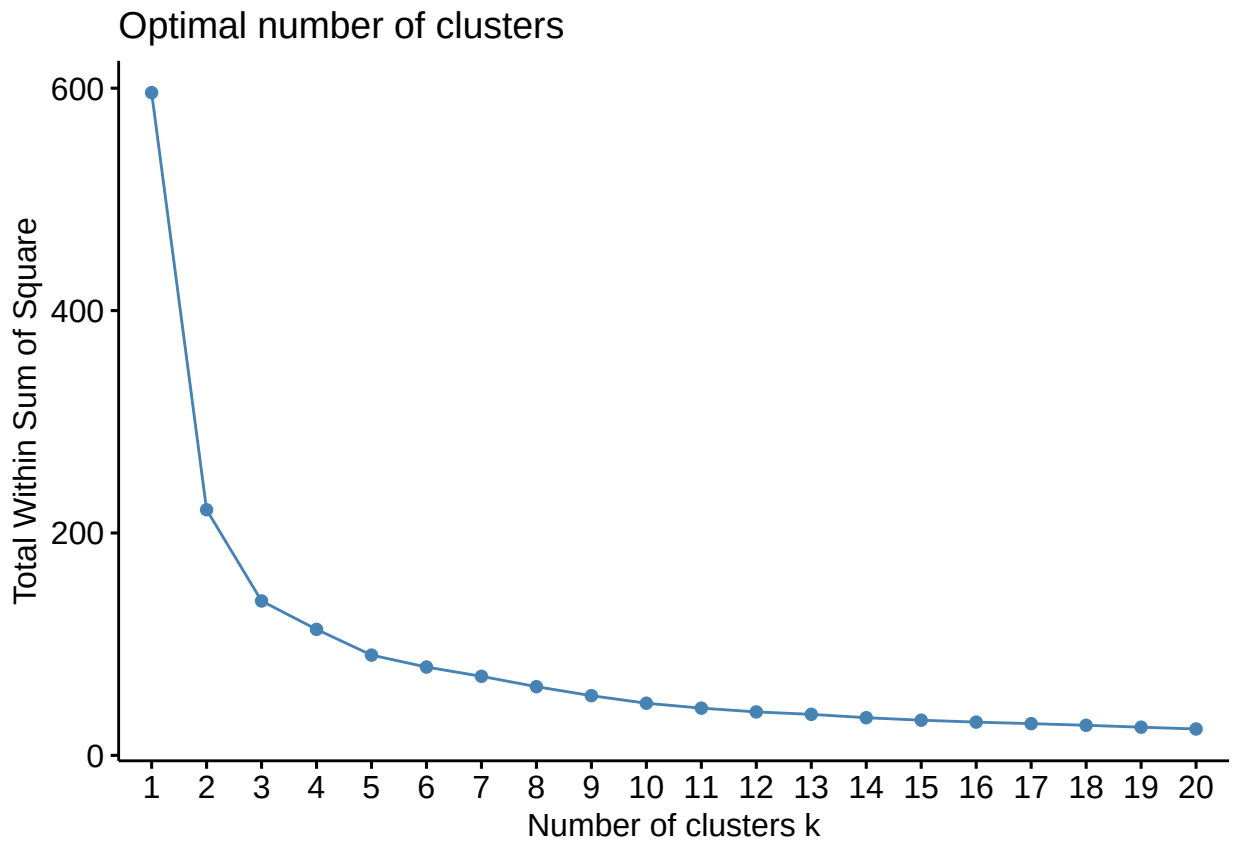

Figura 4: Sumas intra-clusters para determinar la K óptima

Se puede apreciar como en los valores de $K = 4$ o $K = 5$ la varianza intra-clusters ya no se reduce de manera tan significativa, incluso desde el valor $K = 3$ puede notarse que los subsecuentes valores de K ya no reducen tan significativamente a la varianza intra-clusters.

En resumen, un número óptimo de clusters sobre la información de las variables del ancho y largo de los pétalos y sépalos en la tabla iris (4 variables) es 3, a lo más 4 o 5 (mientras menos clusters, mejor).

A continuación se obtienen los clusters.

```
agrupamiento2 <- kmeans(datos, centers = 3, nstart = 50) # 4 variables
```

Y a partir de dicho agrupamiento de la información en 3 clusters puede graficarse en 2 dimensiones a través de un **análisis de componentes principales**, es importante entender que dicho análisis es solamente una transformación lineal de los datos para poder representarlos de forma aproximada a través de proyecciones, dicho análisis **no es una reducción de dimensionalidad del problema**.

```r
fviz_cluster(object = agrupamiento2, data = datos, show.clust.cent = TRUE,
             ellipse.type = "euclid", star.plot = TRUE, repel = TRUE) +
  labs(x="Primera componente",y="Segunda componente") +
  theme_bw() +
  theme(legend.position = "none")
```

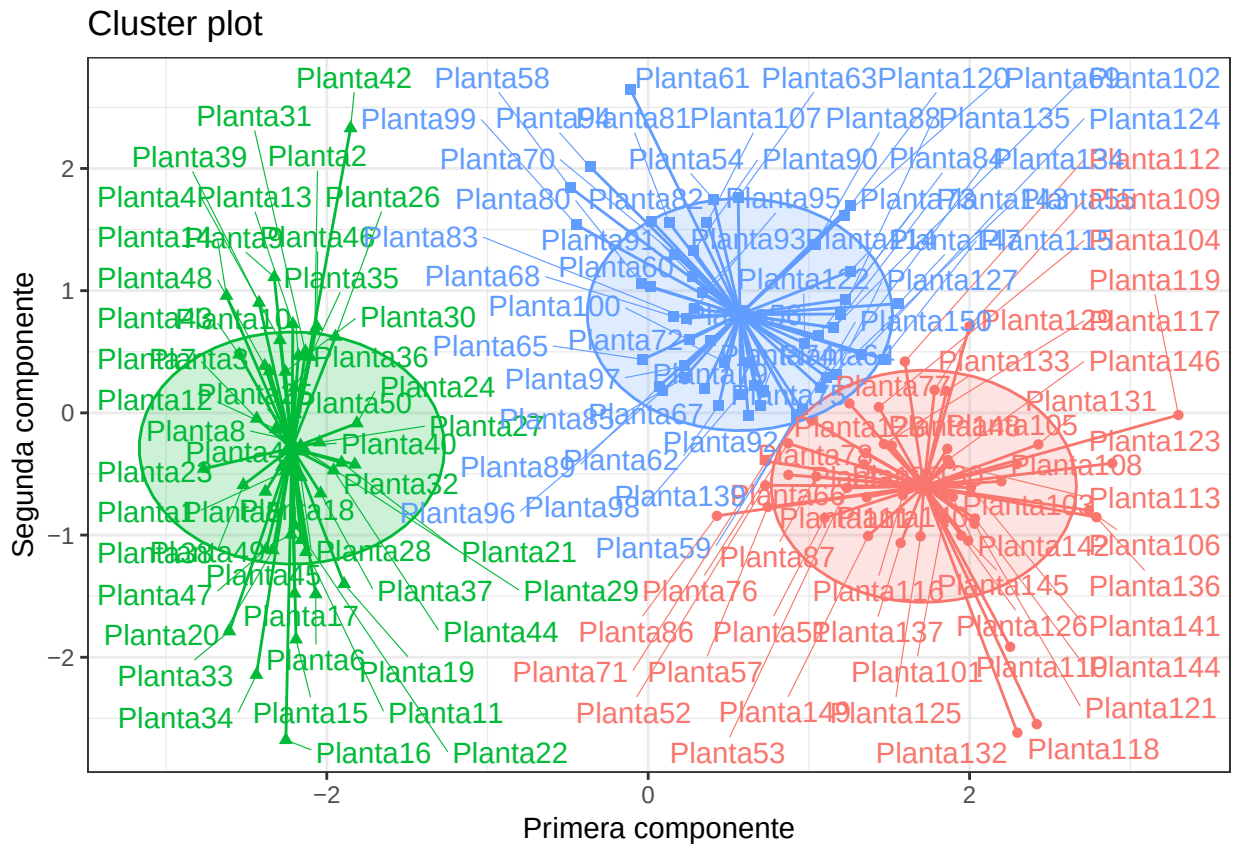

Figura 5: Representación de los clusters a través de análisis de componentes principales

Caso práctico - Pronóstico de ventas

Una importante aplicación del clustering es **el pronóstico de ventas**.

El pronóstico de ventas se resume en realizar una estimación sobre las ventas en cierto periodo futuro que tendrá una tienda, por ejemplo, con base en información histórica.

Múltiples factores pueden afectar dichas ventas futuras: el precio de los bienes, la introducción de productos, el inventario actual, la demanda, entre otros.

[Continuará...]