Estadística Aplicada 3 - Tarea 1

Marcelino

2023-10-16

```
#Cargamos paquetes
library(tidymodels)
library(discrim)
library(corrr)
library(paletteer)
library(MASS)
library(dslabs)
library(tidyr)

# Cargamos bases de datos
data2 <- iris
```

1.- Derive la probabilidad de mala clasificación para LDA

La probabilidad de mala clasificación binaria en LDA (guassiano) es

$$P(\Delta) = P(\mathbf{x} \in R_2 | \mathbf{x} \in \Pi_1)\pi_1 + P(\mathbf{x} \in R_1 | \mathbf{x} \in \Pi_2)\pi_2$$

donde tenemos que

$$P(\mathbf{x} \in R_2 | \mathbf{x} \in \Pi_1)\pi_1 = P(L(\mathbf{x}) < 0 | \mathbf{x} \in \Pi_1)$$

Donde aquí consideramos que $L(\mathbf{x})$ es la función discriminante de LDA, R_2 es la región de decisión para la clase 2, y π_1 ,π_2 son las probabilidades a priori de las clases 1 y 2 respectivamente:

$$L(\mathbf{x}) = \ln\left(\frac{f_1(x)\pi_1}{f_2(x)\pi_2}\right) = \ln\left(\frac{f_1(x)}{f_2(x)}\right) + \ln\left(\frac{\pi_1}{\pi_2}\right)$$

Con lo cual desarrollando (usando propiedades de logaritmo) obtenemos la siguiente expresión:

$$L(\mathbf{x}) = \ln(f_1(x)) - \ln(f_2(x)) + \ln\left(\frac{\pi_1}{\pi_2}\right) =$$

$$\ln(f_1(x)) - \ln(f_2(x)) + \ln\left(\frac{\pi_1}{\pi_2}\right) =$$

$$\ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right)\right)$$

$$-\ln\left(\frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)\right)\right) + \ln\left(\frac{\pi_1}{\pi_2}\right) =$$

$$\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right) -$$

$$\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)\right) + \ln\left(\frac{\pi_1}{\pi_2}\right) =$$

A parti de aquí aplicamos las propiedades de simetría de Σ y además con el desarrollo algebraico obtenemos:

$$L(\mathbf{x}) = \left(-\frac{1}{2}\left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_2^T\Sigma^{-1}\mu_2\right) + \ln\left(\frac{\pi_1}{\pi_2}\right)\right) +$$

$$\mu_1^T\Sigma^{-1}\mathbf{x} - \mu_2^T\Sigma^{-1}\mathbf{x}$$

Con lo cual podemos reescribir la expresión como:

$$L(\mathbf{x}) = b_0 + \mathbf{b}'\mathbf{x}$$

donde

$$b_0 = -\frac{1}{2}\left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_2^T\Sigma^{-1}\mu_2\right) + \ln\left(\frac{\pi_1}{\pi_2}\right)$$

$$\mathbf{b} = \Sigma^{-1}(\mu_1 - \mu_2)$$

Ahora volviendo a lo que nos concierne, considerando la probabilidad condicional, tendríamos la siguiente igualdad:

$$P(L(\mathbf{x}) < 0 | \mathbf{x} \in \Pi_1) = P_{\Pi_1}(b_0 + \mathbf{b}'\mathbf{x} < 0) =$$

Ahora estandarizamos la expresión:

$$P_{\Pi_1}\left(\frac{\mathbf{b}'\mathbf{x} - \mathbf{b}'\mu_1}{\sqrt{(b'\Sigma b)}} < -\frac{(b_0 + \mathbf{b}'\mu_1)}{\sqrt{b'\Sigma b}}\right) = \Phi\left(-\frac{(b_0 + \mathbf{b}'\mu_1)}{\sqrt{b'\Sigma b}}\right)$$

y análogamente para la otra probabilidad que tendríamos que calcular tendríamos:

$$P(\mathbf{x} \in R_1 | \mathbf{x} \in \Pi_2) = \Phi\left(\frac{(b_0 + \mathbf{b}'\mu_2)}{\sqrt{b'\Sigma b}}\right)$$

Con lo cual la probabilidad de error de clasificación es el siguiente:

$$P(\Delta) = \Phi\left(-\frac{(b_0 + \mathbf{b}'\mu_1)}{\sqrt{b'\Sigma b}}\right)\pi_1 + \Phi\left(\frac{(b_0 + \mathbf{b}'\mu_2)}{\sqrt{b'\Sigma b}}\right)\pi_2$$

2. **Implementar LDA sobre la base de datos `MNIST` y usar los dígitos 1 y 3**

La implementación en R fue usando la librería de `MASS` para el método de LDA, y la librería `dslabs` para acceder a los datos de MNIST

Se inicializan los datos de la siguiente forma:

```
mnist_data <- read_mnist()
```

Los datos ya vienen divididos en variables `train` y `test`, las que a su vez contienen `images` y `labels`.

Ahora, los datos de imágenes 28x28 vienen en forma de vectores de 784 entradas, con valores del 0 al 255 representando la gamma del gris. Debido a su fuerte correlación entre pixeles adyacentes, es importante reducir la colinearidad al tomar en vez un estadístico de los pixeles que nos ayude todavía a diferenciarlos. Para esto, decidimos tomar el promedio de los renglones de la imagen para determinar cuánto está dibujado de negro cada fila en la imagen; un dato que debe ayudarnos en general a distinguir 1s de 3s.

```
#Extraer el train y test
tri <- train_images <- mnist_data$train$images
train_labels<-mnist_data$train$labels
tei <- test_images <- mnist_data$test$images
```

```r
test_labels <- mnist_data$test$labels

#Transformar el train al estadístico que deseamos de solo 1 y 3
tr_ind <- (train_labels == 1) | (train_labels==3)
train_labels <- train_labels[tr_ind]
tri <- tri[tr_ind,]
tri2 <- array(tri, dim=c(dim(tri)[1],28,28))
tri_promr <- rowMeans(tri2, dims = 2)

#Transformar el test al estadístico que deseamos de solo 1 y 3
te_ind <- (test_labels == 1) | (test_labels==3)
test_labels <- test_labels[te_ind]
tei <- tei[te_ind,]
tei2 <- array(tei, dim=c(dim(tei)[1],28,28))
tei_promr <- rowMeans(tei2, dims = 2)
```

Una vez hecho eso, ahora aplicamos el algoritmo de LDA para crear el predictor, construyendo la matriz (Y,X) para insertar al modelo.

```r
train <- data.frame(label = (train_labels == 1),tri_promr)

model <- lda(label~.,data=train)
```

Una vez entrenado, ahora lo ponemos a prueba con el set de `test`, y medimos la precisión.

```r
test <- data.frame(label = (test_labels == 1),tei_promr)


prob <- predict(model, test, method = "debiased") #, type = "response")
pred <- ifelse(prob$posterior[,2] > 0.5, 1, 3)

sum(pred == test_labels) / length(test_labels) #precisión de predictor
```

```
[1] 0.8899767
```

Con ello, obtenemos al final los resultados del experimento, lo cual nos da:

```r
prob <- predict(model, test, method = "debiased") #, type = "response")
pred <- ifelse(prob$posterior[,2] > 0.5, 1, 3)
sum(pred == test_labels) / length(test_labels) #precisión de predictor
```

```
[1] 0.8899767
```

Eso nos da una precisión del 88.99%, bastante alto para un predictor lineal de clases entre 1 y 3. Con ello, se conluye el problema.

3.- Implementar QDA sobre los datos de `iris` (raw data)

Vamos a implementar QDA en los datos de `iris` utilizando datos de entrenamiento y testeo dado que no queremos causar un sobreajuste en los datos.

Utilizaremos el 75% de los datos para entranamiento y la semilla 1087.

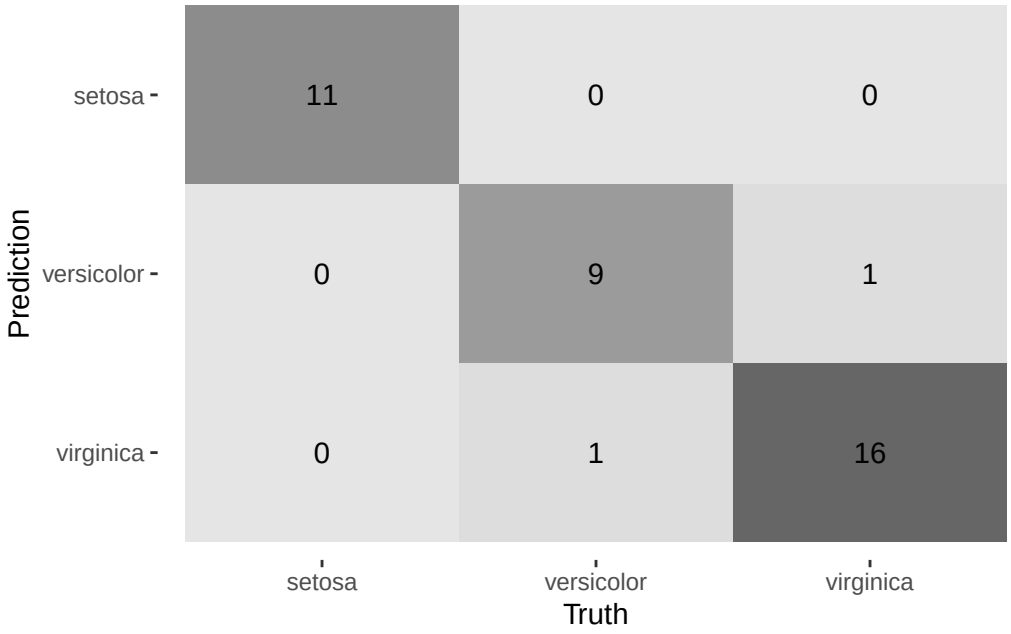
```
set.seed(1087)
data_split <- rsample::initial_split(data2, prop = .75)
data_train <- training(data_split)
data_test <- testing(data_split)

qda_spec <- discrim_quad() |>
  set_mode("classification") |>
  set_engine("MASS")

qda_fit <- qda_spec |>
  fit(Species ~ ., data = data_train)
```

Ahora calcularemos métricas para saber qué tan bien estuvo nuestro ajuste.

```
augment(qda_fit, new_data = data_test) |>
  conf_mat(truth = Species, estimate = .pred_class) |>
  autoplot(type = "heatmap")
```




```
augment(qda_fit, new_data = data_test) |>
  accuracy(truth = Species, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy multiclass     0.947
```

```
augment(qda_fit, new_data = data_test) |>
  recall(truth = Species, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 recall  macro          0.947
```

```
augment(qda_fit, new_data = data_test) |>
  precision(truth = Species, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision macro          0.947
```

Con lo cual notamos que tuvimos un muy buen ajuste, dado que entre más cercano a uno, las métricas nos dicen que nuestro ajuste será muy bueno para predecir nuevos datos.

Por último veremos las estadísticas de nuestro ajuste

```
qda_fit$fit
```

```
Call:
qda(Species ~ ., data = data)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3482143  0.3571429  0.2946429

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa         5.023077    3.423077     1.471795   0.2384615
versicolor     5.915000    2.750000     4.237500   1.3150000
virginica      6.539394    2.957576     5.518182   2.0575758
```

Con lo cual notamos que en un principio habían casi la misma proporción de especies en la base de datos.

4. **Problema 8.3 Izenman**

Tenemos $X_1 \sim N_r(\mu_1, \Sigma_{XX}), X_2 \sim N_r(\mu_2, \Sigma_{XX})$ independientes, y tenemos el siguiente estadístico:

$$\frac{\{\mathbb{E}[a^T X_1] - \mathbb{E}[a^T X_2]\}^2}{Var(a^T X_1 - a^T X_2)}$$

Ahora, si consideramos $X_1 - X_2 \sim N_r(\mu_1 - \mu_2, 2\Sigma_{XX}) =: N_r(\mu_Y, \Sigma_{YY})$. Además, tomando en cuenta la norma definida por una matriz. Por lo tanto tenemos

$$\max_{a \in \mathbb{R}^r} G(a) = (a^T \mu_Y)^2, s.a. \quad a^T \Sigma_{YY} a = 1$$

Para esto, empezamos derivando el multiplicador:

$$F(a) = (a^T \mu_Y)^2 - \lambda a^T \Sigma_{YY} a$$
$$\implies \partial_a F(a) = 2(\mu_Y \mu_Y^T - \lambda \Sigma_{YY})a = 0$$
$$\implies (\mu_Y \mu_Y^T - \lambda \Sigma_{YY})a = 0 \implies a(\mu_Y \mu_Y^T - \lambda \Sigma_{YY})a = 0 \implies (a^T \mu_Y)^2 = \lambda$$
$$\therefore \sqrt{\lambda}\mu_Y - \lambda \Sigma_{YY} a = 0 \therefore a = \frac{1}{\sqrt{\lambda}} \Sigma_{YY}^{-1} \mu_Y \quad \alpha \quad \Sigma_{XX}^{-1}(\mu_1 - \mu_2)$$

Con esto, concluímos el problema.

5.- Problema de clasificación

A reseracher wants to determine a prcedure for disminating between two multivariate populations. The researcher has enough datavailable to estimate the density functions $f_1(x)$ and $f_2(x)$ associated with populatios π_1 and π_2 respectively. Let $c(2|1) = 50$ (cost of assigning item as π_2 given that $\pi_1 istrue$) and $c(2|1) = 100$

In addition, it is knoe that about 20% of all possible items (for which the measurements x can be recorded) belong to π_2

(a) Give the minimum ECM rule (in general form) for assigning a new item to one of the two populations.

(b) Measurements recorded on a new item yield the densitiy values $f_1(x) = .3$ and $f_2(x) = .5$. Given the preceding information, assign this item to populatio π_1 or π_2

Para el primer inciso, tenemos la regla mínima del ECM:

$$R_1 : \frac{f_1(x)}{f_2(x)} \geq \frac{\pi_2}{\pi_1} \frac{c(1|2)}{c(2|1)}; \quad R_2 : \frac{f_1(x)}{f_2(x)} < \frac{\pi_2}{\pi_1} \frac{c(1|2)}{c(2|1)}$$

El problema ya nos dice que π_2=0.2, al igual que nos da los valores de $c(1,2) = 100$ y $c(2,1) = 50$, por lo cual reescribimos:

$$R_1 : \frac{f_1(x)}{f_2(x)} \geq 0.5; \quad R_2 : \frac{f_1(x)}{f_2(x)} < 0.5$$

Para el segundo inciso, tenemos $f_1(x) = .3$ y $f_2(x) = .5$. Podemos estimar por tanto la regla mínima:

$$\frac{f_1(x)}{f_2(x)} = 0.6 \geq 0.5$$

Con esto podemos concluir que el nuestro objeto **tiene que clasificarse en Π_1.**

6.- Problema de QDA

Suppose x comes from one of two populations:

$$\pi_1 \sim N(\mu_1, \Sigma_1)$$

$$\pi_2 \sim N(\mu_2, \Sigma_2)$$

If the respective density functions are denoted by f_{1}(x) and f_{2}(x), find the expression for the quadratic disccriminator

$$Q = \ln\left(\frac{f_1(x)}{f_2(x)}\right)$$

If $\Sigma_1 = \Sigma_2 = \Sigma$ for instance, verify that Q becomes

$$(\mu_1 - \mu_2)'\Sigma^{-1}x - \frac{1}{2}(\mu_1 - \mu_2)'\Sigma^{-1}(\mu_1 + \mu_2)$$

Tenemos que las densidades para una variable normal r-variada se describe como:

$$f(x) = \frac{1}{(2\pi)^{r/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

Por tanto, eso nos da:

$$\begin{aligned} Q &= \ln\left(\frac{\frac{1}{(2\pi)^{r/2}|\Sigma_1|^{1/2}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}}{\frac{1}{(2\pi)^{r/2}|\Sigma_2|^{1/2}} e^{-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}\right) \\ &= \ln\left(\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} e^{\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}\right) \\ &= \ln\left(\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}}\right) + \frac{1}{2}\left\{(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right\} \end{aligned}$$

Ahora, bajo el supuesto que $\Sigma_1 = \Sigma_2 = \Sigma$, obtenemos:

$$= \ln\left(\frac{|\Sigma|^{1/2}}{|\Sigma|^{1/2}}\right) + \frac{1}{2}\left\{(x-\mu_2)^T\Sigma^{-1}(x-\mu_2) - (x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}$$

$$= \ln(1) + \frac{1}{2}\left\{x^T\Sigma^{-1}x - \mu_2^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_2 + \mu_2^T\Sigma^{-1}\mu_2\right\}$$

$$-\frac{1}{2}\left\{x^T\Sigma^{-1}x - \mu_1^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_1 + \mu_1^T\Sigma^{-1}\mu_1\right\}$$

$$= \frac{1}{2}\left\{(\mu_1-\mu_2)^T\Sigma^{-1}x + x^T\Sigma^{-1}(\mu_1-\mu_2) - (\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1+\mu_2)\right\}$$

$$= (\mu_1-\mu_2)^T\Sigma^{-1}x - \frac{1}{2}(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1+\mu_2)$$

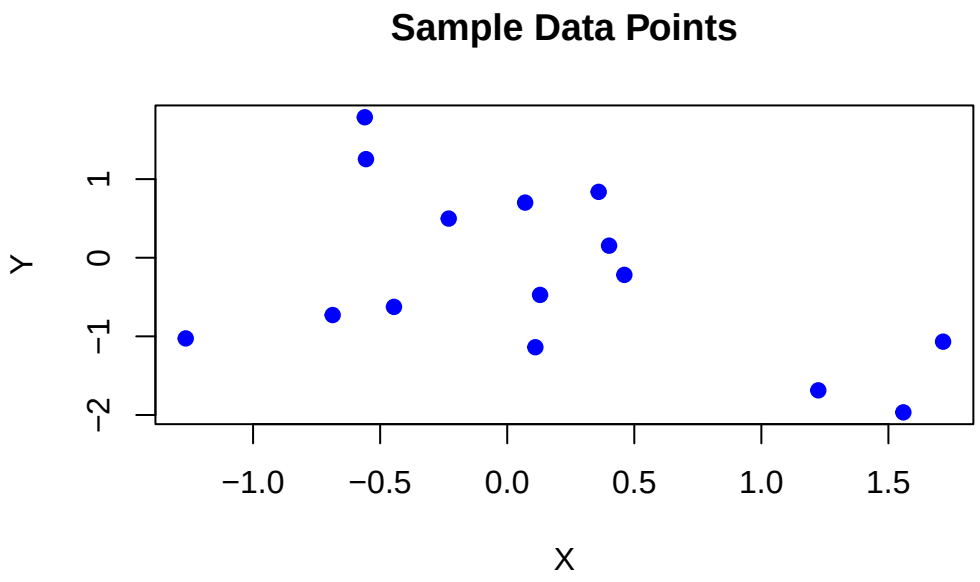
Con esto, concluímos el problema.

7.- Write a coputer program to implement single-linkage, average-linkage, and complete-linkage agglomerative hierarchical clustering. Try it out on a data set of your choice.

Creamos una base de datos aleatoria primero y después contruimos la matriz de disimalaridad con la distancia euclidiana.

```
# Sample data
set.seed(123)
data <- matrix(rnorm(30), ncol=2)
colnames(data) <- c("X", "Y")

# Visualize the data
plot(data, pch=19, col="blue", xlab="X", ylab="Y", main="Sample Data Points")
```



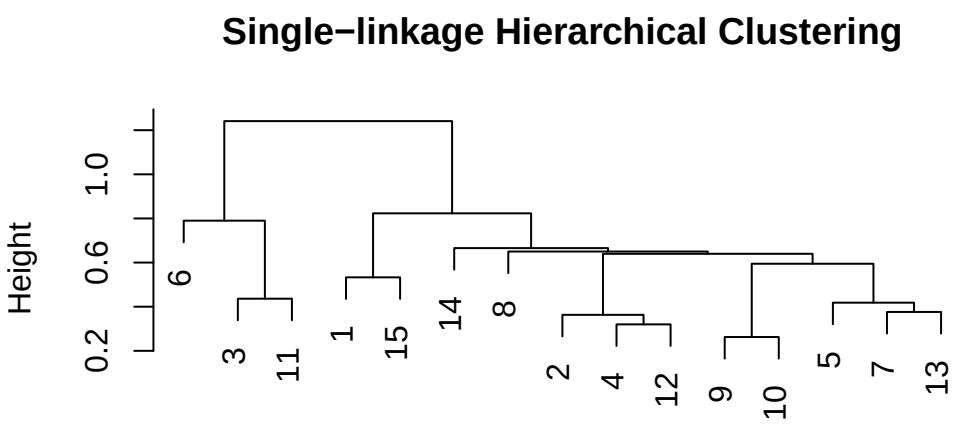

```
dist_matrix <- dist(data)
```

Ahora procedemos a crear el programa ayudado con un paquete de R que ya lo calcula con la función de **hclust** del paquete **stats**. Y programamos para que directamente grafique los dendogramas.

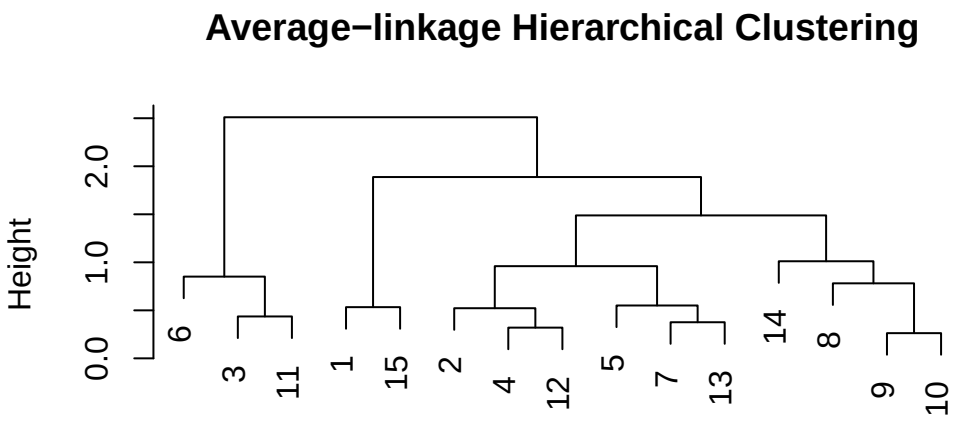
```
clusterFunc <- function(dist_matrix,method ="single" ){
  if(method=="single"){
    # Single-linkage
    single_linkage <- hclust(dist_matrix, method="single")
    plot(single_linkage, main="Single-linkage Hierarchical Clustering", sub="", xlab="", y
  } else if(method=="average"){
    # Average-linkage
    average_linkage <- hclust(dist_matrix, method="average")
    plot(average_linkage, main="Average-linkage Hierarchical Clustering", sub="", xlab="",
  } else{
    # Complete-linkage
    complete_linkage <- hclust(dist_matrix, method="complete")
    plot(complete_linkage, main="Complete-linkage Hierarchical Clustering", sub="", xlab="
  }
}
```

Y procedemos a probarlo

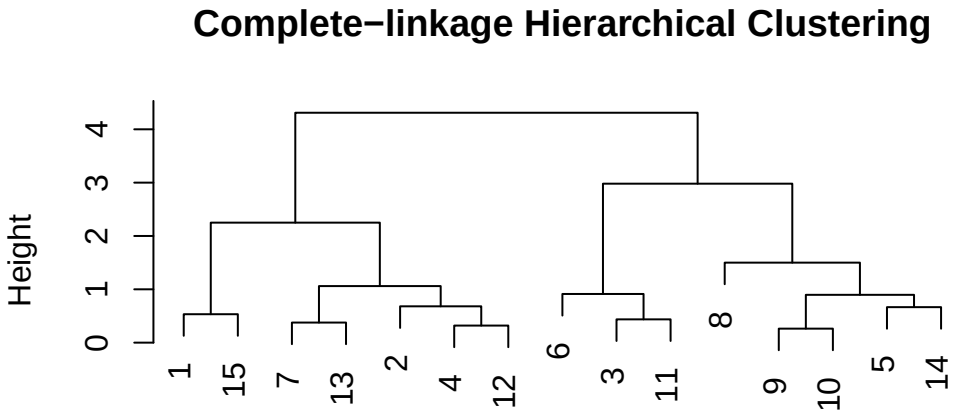
```
clusterFunc(dist_matrix,method ="single" )
```




```
clusterFunc(dist_matrix,method ="average" )
```




```
clusterFunc(dist_matrix,method ="complete" )
```



8.- Implemente SL, AL y CL sobre la base de datos `iris` (raw data)

Primero inicializamos `iris` y agarramos los datos numéricos de su base de datos

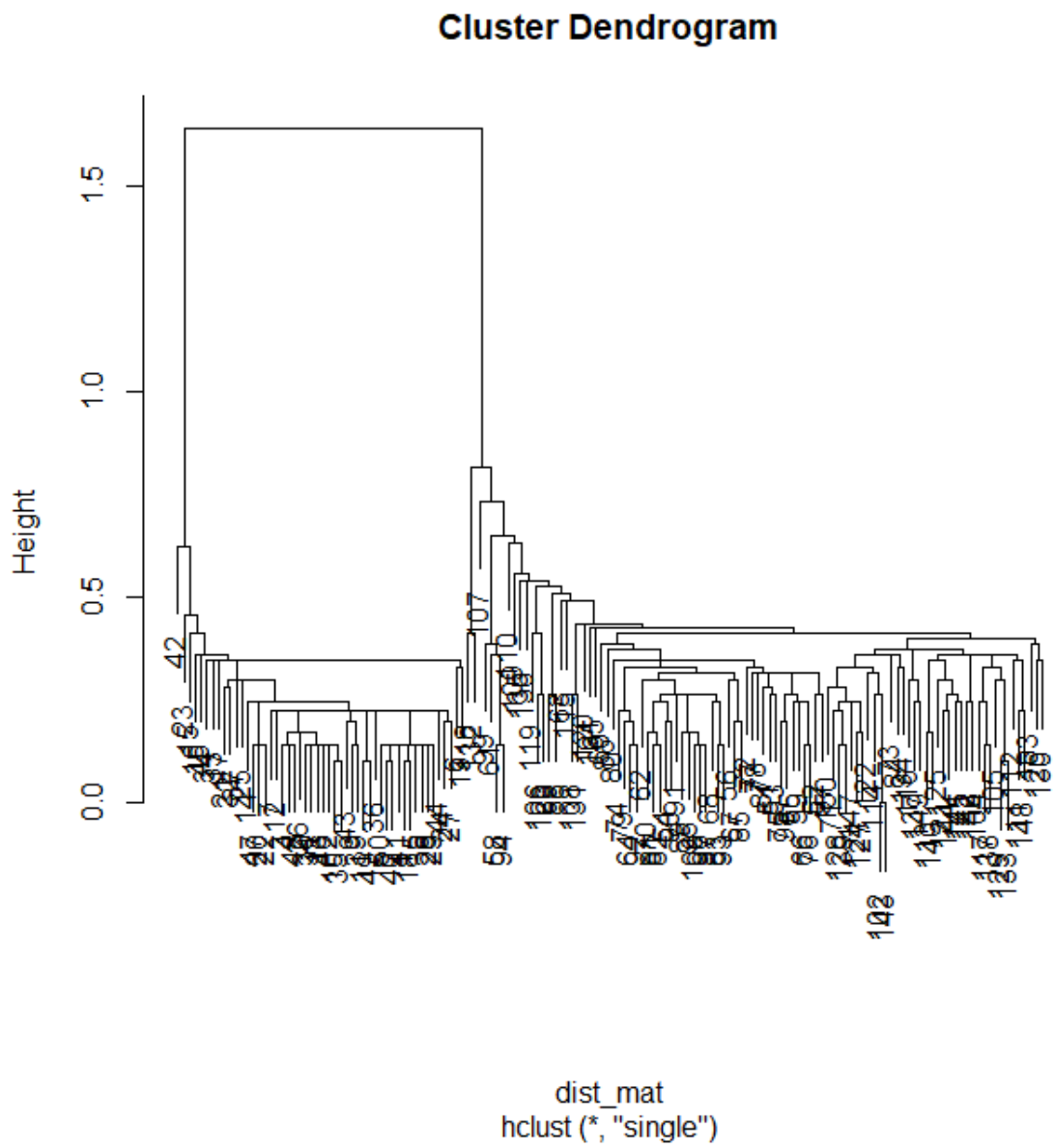
```
library(cluster)

#Iris DB
data <- as.matrix(iris[,1:4])
dist_mat <- dist(data, method = 'euclidean')
```

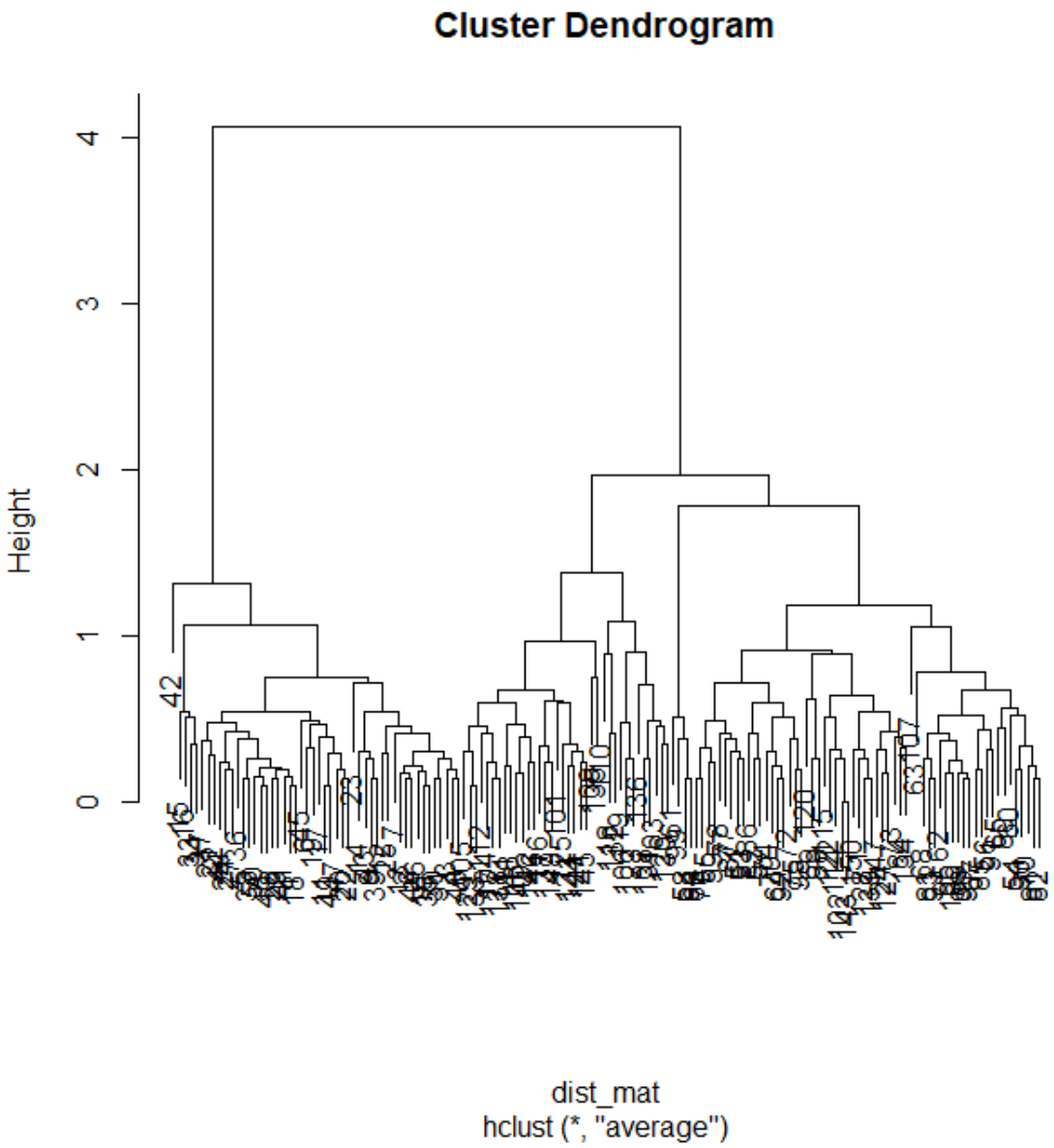
Luego, usamos clustering aglomerativo usando `hclust` y clustering divisivo usando `agnes`, con ello corremos lo siguiente:

```
hclust_single <- hclust(dist_mat, method = 'single')
hclust_average <- hclust(dist_mat, method='average')
hclust_complete <- hclust(dist_mat, method= 'complete')
divisive_model <- agnes(dist_mat, method = "single")
```

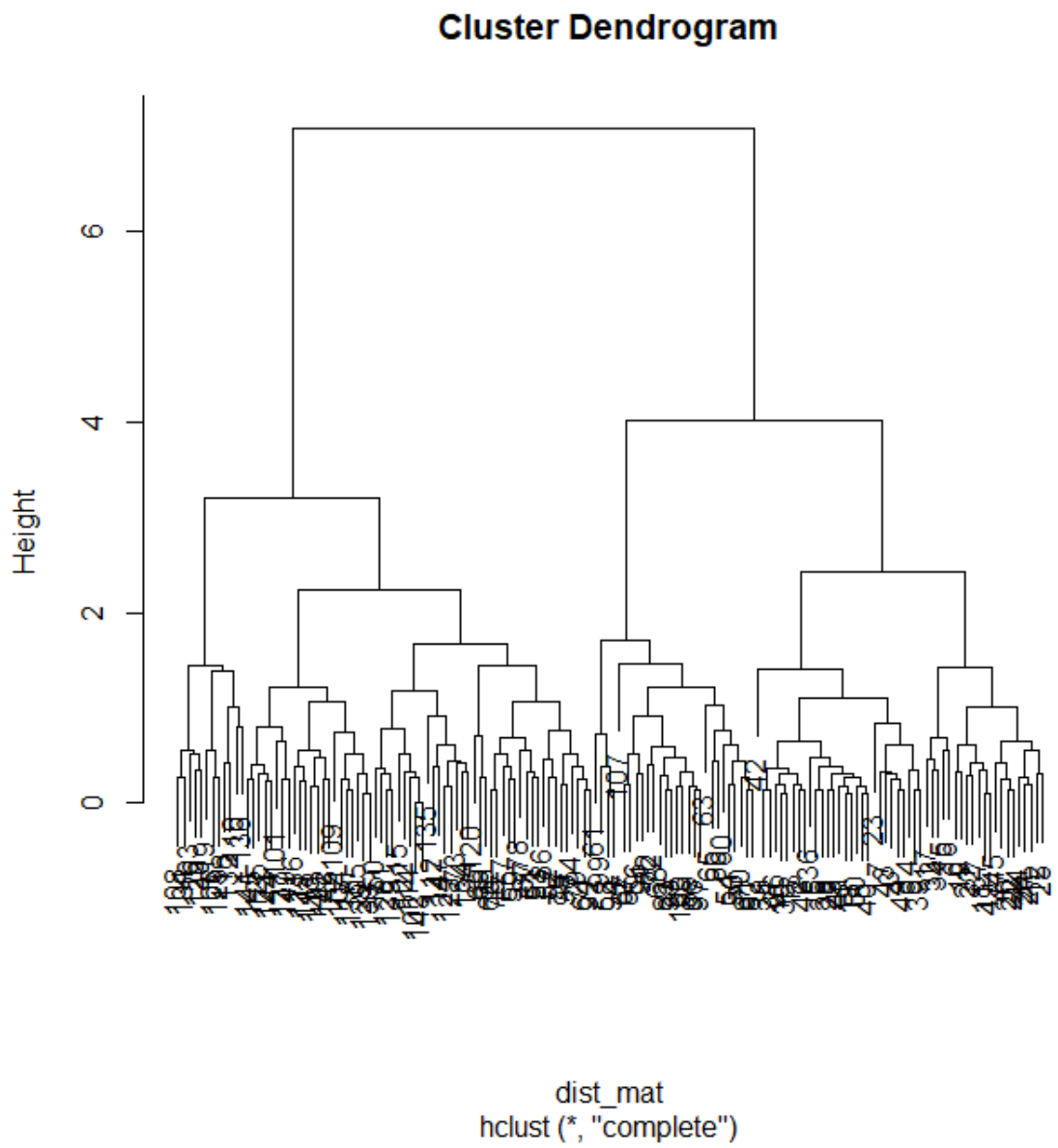
Los plots resultantes son los siguientes:



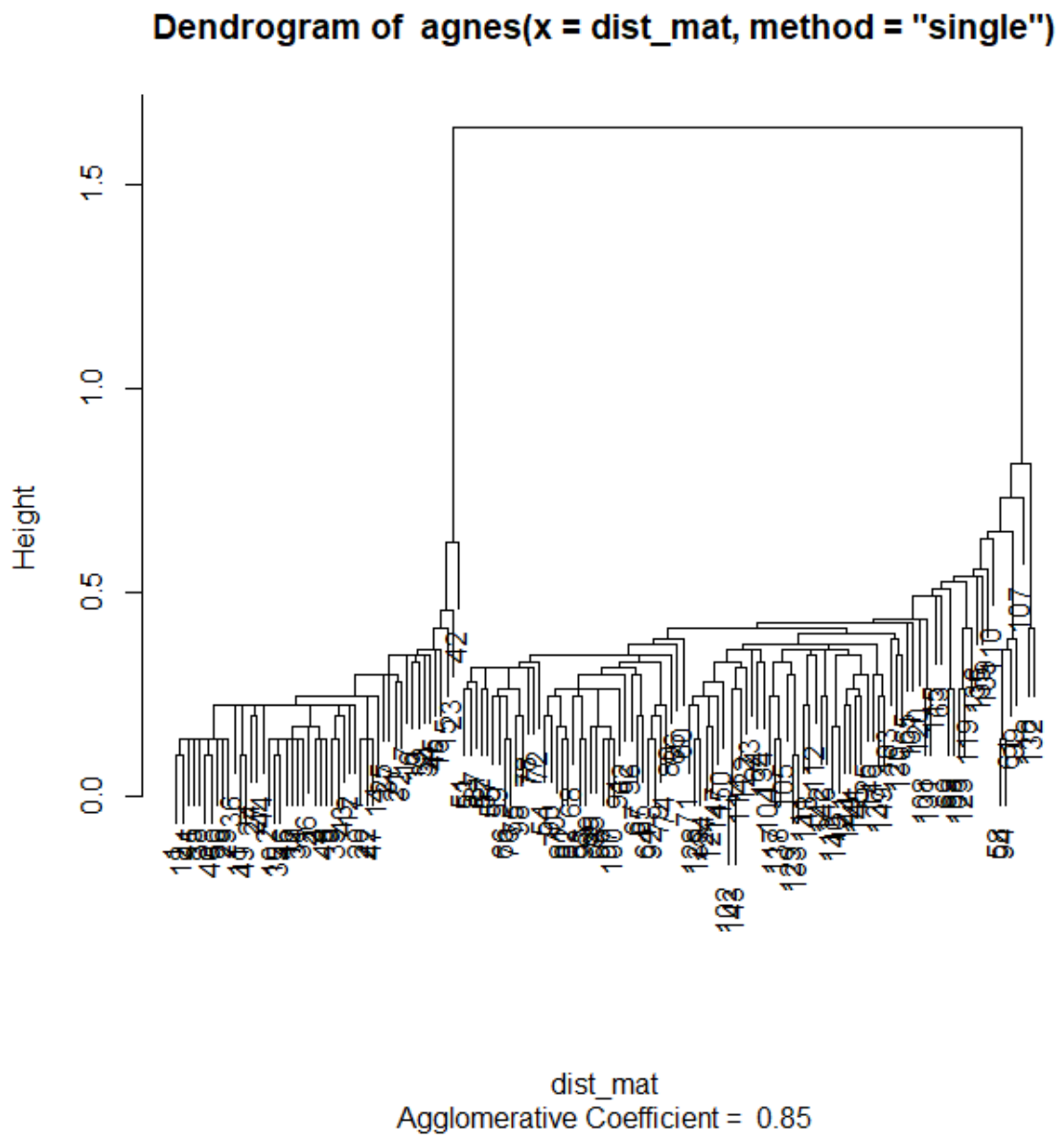
Single Linkage Agglomerative Cluster



Average Linkage Agglomerative Cluster



Complete Linkage Agglomerative Cluster



Divisive Cluster

De estas imágenes podemos determinar, junto con el conocimiento previo de que hay **tres** tipos de plantas, que el Complete Linkage Cluster es el que mejor nos permite diferenciar los tipos de planta que tiene la base de datos al hacer un corte horizontal a altura 3.5, mientras que los demás algoritmos tienen problema diferenciando los tipos de especies al tomar diferencias mínimas o promedio de medidas en sépalo y pétalo. Por tanto, **es más fácil diferenciar las especies de plantas por el tamaño máximo de sépalo y pétalo de las plantas, haciendo el Complete Linkage el más efectivo.**

Cereal	Protein_gm	Carbohydrates_gm	Fat_gm	Calories_per_oz	Vitamin_A_pct_dail
Life	6	19	1	110	
Grape Nuts	3	23	0	100	
Super Sugar Crisp	2	26	0	110	
Special K	6	21	0	110	
Rice Krispies	2	25	0	110	
Raisin Bran	3	28	1	120	
Product 19	2	24	0	110	
Wheaties	3	23	1	110	
Total	3	23	1	110	
Puffed Rice	1	13	0	50	
Sugar Corn Pops	1	26	0	110	
Sugar Smacks	2	25	0	110	

9.- Problem 12.13

The following table lists measurements on 5 nutritional variables for 12 breakfast cereals.

```
library(kableExtra)
cereal_data <- data.frame(
  Cereal = c("Life", "Grape Nuts", "Super Sugar Crisp", "Special K", "Rice Krispies", "Rai
             "Product 19", "Wheaties", "Total", "Puffed Rice", "Sugar Corn Pops", "Sugar S
  Protein_gm = c(6, 3, 2, 6, 2, 3, 2, 3, 3, 1, 1, 2),
  Carbohydrates_gm = c(19, 23, 26, 21, 25, 28, 24, 23, 23, 13, 26, 25),
  Fat_gm = c(1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0),
  Calories_per_oz = c(110, 100, 110, 110, 110, 120, 110, 110, 110, 50, 110, 110),
  Vitamin_A_pct_daily_allowance = c(0, 25, 25, 25, 25, 25, 100, 25, 100, 0, 25, 25)
)

kable(cereal_data, "latex", booktabs = TRUE) %>%
  kable_styling()
```

(a) Using tha data in the table, calculate the euclidean distances between each pair of cereal brands.

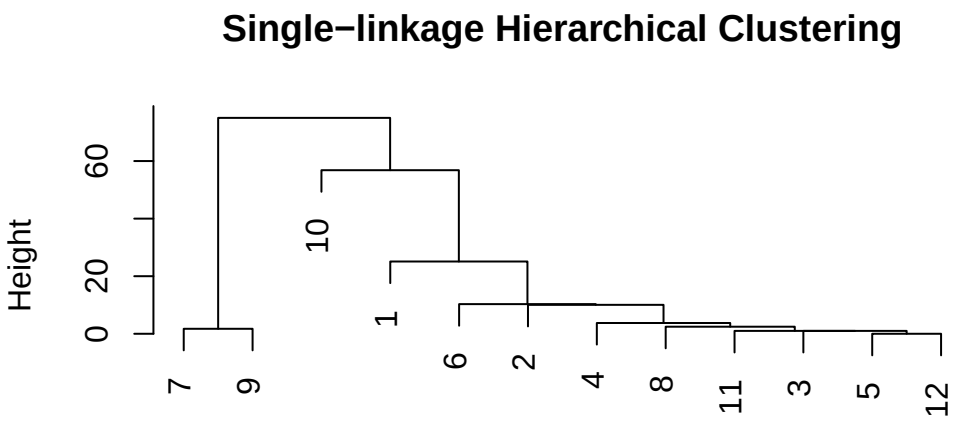
```
# Compute the distance matrix
dist_matrix <- dist(cereal_data[1:12,2:6])

kable(as.matrix(dist_matrix),"latex",booktabs=TRUE) %>% kable_styling()
```

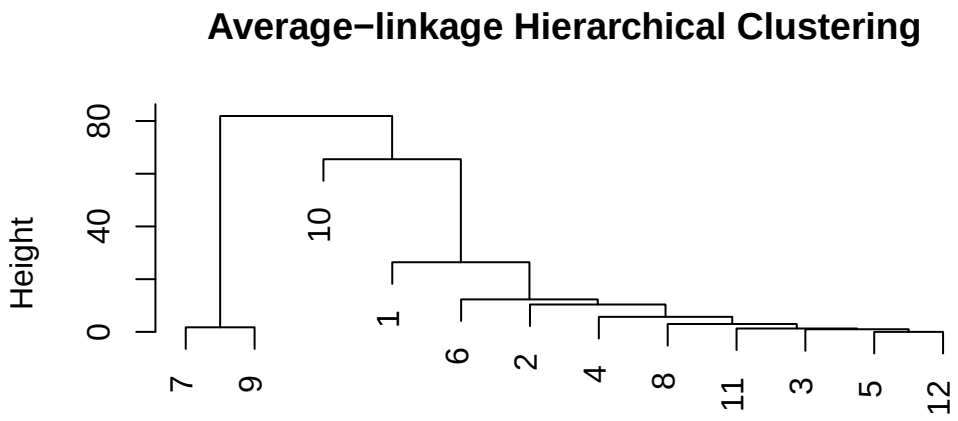
1	2	3	4	5	6	7	8	9
0.00000	27.40438	26.286879	25.099801	26.038433	28.54820	100.209780	25.495098	100.124922
27.40438	0.00000	10.488088	10.630146	10.246951	20.63977	75.676945	10.049876	75.670338
26.28688	10.48809	0.000000	6.403124	1.000000	10.29563	75.026662	3.316625	75.073297
25.09980	10.63015	6.403124	0.000000	5.656854	12.60952	75.166482	3.741657	75.093275
26.03843	10.24695	1.000000	5.656854	0.000000	10.53565	75.006666	2.449490	75.039989
28.54820	20.63977	10.295630	12.609520	10.535654	0.00000	75.782584	11.180340	75.828754
100.20978	75.67695	75.026662	75.166482	75.006666	75.78258	0.000000	75.019997	1.732051
25.49510	10.04988	3.316625	3.741657	2.449490	11.18034	75.019997	0.000000	75.000000
100.12492	75.67034	75.073297	75.093275	75.039989	75.82875	1.732051	75.000000	0.000000
60.51446	56.82429	66.294796	65.681048	66.105975	75.86172	117.140941	65.802735	117.068356
26.45751	10.63015	1.000000	7.071068	1.414214	10.44031	75.033326	3.741657	75.093275
26.03843	10.24695	1.000000	5.656854	0.000000	10.53565	75.006666	2.449490	75.039989

(b) Treating the distances calculated in (a) as measures of dissimilarity, use single-linkage, complete-linkage, and average-linkage clustering to cluster the cereal brands. Construct a dendrogram for each method.

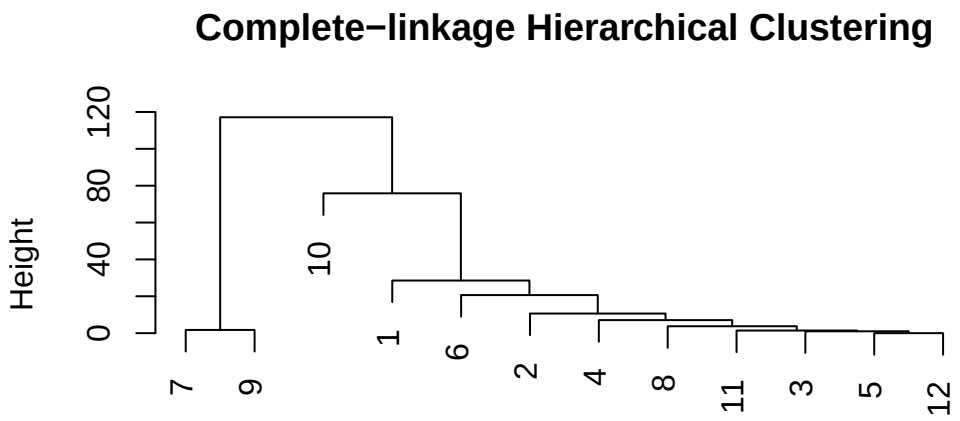
```
clusterFunc(dist_matrix,method ="single" )
```



```
clusterFunc(dist_matrix,method ="average" )
```




```
clusterFunc(dist_matrix,method ="complete" )
```



De estas imágenes podemos determinar de que hay casi **tres** tipos de cereales. Y en los 3 métodos los dendogramas se parecen muchísimo. Con lo cual es mucho más fácil concordar en los distintos clusters que se pueden hacer a diferencia del anterior.

10.- Input the data in the previous table into k-means clustering program. Use the program to cluster the cereal brands into 2, 3, and 4 clusters. Compare the results with those obtained in the previous problem.

Utilizamos K=2

```
# Exclude the Cereal column
clustering_data <- cereal_data[,-1]

# Choose the optimal number of clusters (for demonstration, let's say 3)
set.seed(123)  # Set seed for reproducibility
clusters <- kmeans(clustering_data, centers=2, nstart=20)

# Add cluster results to the original data
```

Cereal	Protein_gm	Carbohydrates_gm	Fat_gm	Calories_per_oz	Vitamin_A_pct_dail
Life	6	19	1	110	
Grape Nuts	3	23	0	100	
Super Sugar Crisp	2	26	0	110	
Special K	6	21	0	110	
Rice Krispies	2	25	0	110	
Raisin Bran	3	28	1	120	
Product 19	2	24	0	110	
Wheaties	3	23	1	110	
Total	3	23	1	110	
Puffed Rice	1	13	0	50	
Sugar Corn Pops	1	26	0	110	
Sugar Smacks	2	25	0	110	

```
cereal_data$cluster <- as.factor(clusters$cluster)

kable(cereal_data, "latex", booktabs = TRUE) %>%
kable_styling()
```

Utilizamos K=3

```
clusters <- kmeans(clustering_data, centers=3, nstart=20)

# Add cluster results to the original data
cereal_data$cluster <- as.factor(clusters$cluster)

# Print the dataframe with clusters
kable(cereal_data, "latex", booktabs = TRUE) %>%
kable_styling()
```

Utilizamos K=4

```
clusters <- kmeans(clustering_data, centers=4, nstart=20)

# Add cluster results to the original data
cereal_data$cluster <- as.factor(clusters$cluster)
```

Cereal	Protein_gm	Carbohydrates_gm	Fat_gm	Calories_per_oz	Vitamin_A_pct_dail
Life	6	19	1	110	
Grape Nuts	3	23	0	100	
Super Sugar Crisp	2	26	0	110	
Special K	6	21	0	110	
Rice Krispies	2	25	0	110	
Raisin Bran	3	28	1	120	
Product 19	2	24	0	110	
Wheaties	3	23	1	110	
Total	3	23	1	110	
Puffed Rice	1	13	0	50	
Sugar Corn Pops	1	26	0	110	
Sugar Smacks	2	25	0	110	

```
# Print the dataframe with clusters
kable(cereal_data, "latex", booktabs = TRUE) %>%
kable_styling()
```

Notamos que los clusters que se van formando en k-means son los mismos que se formarían si nos vamos desplazando continuamente en la altura a la cual decidimos cortar de cualquiera de los dendogramas. Por lo tanto se llegan a las mismas conclusiones que utilizando los dendogramas y en este caso serían equivalentes.

Cereal	Protein_gm	Carbohydrates_gm	Fat_gm	Calories_per_oz	Vitamin_A_pct_dail
Life	6	19	1	110	
Grape Nuts	3	23	0	100	
Super Sugar Crisp	2	26	0	110	
Special K	6	21	0	110	
Rice Krispies	2	25	0	110	
Raisin Bran	3	28	1	120	
Product 19	2	24	0	110	
Wheaties	3	23	1	110	
Total	3	23	1	110	
Puffed Rice	1	13	0	50	
Sugar Corn Pops	1	26	0	110	
Sugar Smacks	2	25	0	110	