

# Agrupamiento de datos

Clustering

---

Diana L. Giraldo

Febrero 2021

1. Intro
2.  $k$ -means
3. ¿Cómo elegir  $k$ ?
4. Distancias entre observaciones
5.  $k$ -medoids

# Intro

---

# ¿Qué hago?

En mi tesis de doctorado trabajo con información de:

- Neuroimágenes
- Variables clínicas
- Evaluación neuropsicológica

para la caracterización de patrones asociados a la

**enfermedad de Alzheimer,**

la causa más común de demencia.





Antes de la demencia, cuando los pacientes muestran alguna disfunción cognitiva, se diagnostica el

## Deterioro Cognitivo Leve

Mild cognitive impairment (MCI).

Un conjunto de tests neurpsicológicos evalúan las funciones cognitivas en diferentes dominios: memoria, lenguaje, habilidades visuo-espaciales, etc.

## ¿qué vamos a hacer hoy?

Usar\* puntajes por dominio cognitivo para explorar la existencia de distintos **sub-grupos** de pacientes con Deterioro Cognitivo Leve.

\* aka, cacharrear con datos.

## ¿qué vamos a hacer hoy?

Usar\* puntajes por dominio cognitivo para explorar la existencia de distintos **sub-grupos** de pacientes con Deterioro Cognitivo Leve.

Los datos de prueba, código y esta presentación están en

[https://github.com/diagiraldo/RLadies\\_bta\\_2021/](https://github.com/diagiraldo/RLadies_bta_2021/)

\* aka, cacharrear con datos.

# ¿qué vamos a hacer hoy?

Usar\* puntajes por dominio cognitivo para explorar la existencia de distintos **sub-grupos** de pacientes con Deterioro Cognitivo Leve.

Los datos de prueba, código y esta presentación están en

[https://github.com/diagiraldo/RLadies\\_bta\\_2021/](https://github.com/diagiraldo/RLadies_bta_2021/)

En RStudio:

```
A <- read.csv("datos/neuropsycho_data.csv")  
head(A)
```

	id	edad	edu	genero	dx	progresion	t_progresion	t_seguimiento	MEMORY	LANGUAGE	EXECUTIVE	VISUOSPATIAL	ORIENTATION	ATTENTION
1	1	79.3	16	Male	CN	TRUE	24	84	0.1983688	-0.03209976	0.32784032	-0.4572322	-0.5705812	-0.3139389
2	2	89.5	18	Female	CN	TRUE	60	60	-1.1210747	0.29438495	-0.36517101	1.7030370	-0.8349403	0.2254791
3	3	86.8	18	Male	MCI	FALSE	NA	6	-1.3938079	-0.41181495	-0.46242071	-1.1384656	-0.5148650	2.0804932
4	4	77.6	18	Female	CN	FALSE	NA	84	-1.2479705	-0.67085719	-0.34398220	-0.7515376	-0.8615245	-0.2979084
5	5	76.7	14	Male	CN	FALSE	NA	60	-0.5485783	-0.04031867	0.05542082	-0.4822031	-0.9414579	-0.2955286
6	6	82.7	18	Female	CN	FALSE	NA	96	-1.3993788	-0.29124856	-0.21407394	0.5718734	0.1803515	-0.3387569

\* aka, cacharrear con datos.

# Variables

## Puntajes

### ESTANDARIZADOS por dominio:

- MEMORY
- LANGUAGE
- EXECUTIVE
- VISUOSPATIAL
- ORIENTATION
- ATTENTION

## Información demográfica:

- edad
- género
- edu: años de educación.

## Información clínica:

- dx: diagnóstico
  - CN: control
  - MCI: deterioro cognitivo leve
- progresion: diagnóstico de demencia en futuras visitas?
- t\_progresión: meses entre visita y progresión.
- t\_seguimiento: meses de seguimiento a la persona.

Nos interesan los **puntajes por dominio** de pacientes con Deterioro Cognitivo Leve.

Nos interesan los **puntajes por dominio** de pacientes con Deterioro Cognitivo Leve.

En RStudio:

Nos interesan los **puntajes por dominio** de pacientes con Deterioro Cognitivo Leve.

En RStudio:

**Cargar librerías:**

```
library(dplyr)
library(reshape2)
library(ggplot2)
library(cluster)
library(factoextra)
library(NbClust)
```

Nos interesan los **puntajes por dominio** de pacientes con Deterioro Cognitivo Leve.

En RStudio:

**Cargar librerías:**

```
library(dplyr)
library(reshape2)
library(ggplot2)
library(cluster)
library(factoextra)
library(NbClust)
```

**Filtrar datos:**

```
Amci <- filter(A, dx == "MCI")
B <- Amci %>%
  select(MEMORY, LANGUAGE, EXECUTIVE,
         VISUOSPATIAL, ORIENTATION, ATTENTION)
```

la tabla B tiene 680 observaciones (filas) con 6 variables (columnas)

## Agrupamiento de datos (Clustering)

Conjunto de métodos **no-supervisados** para dividir un grupo de datos en sub-grupos de tal forma que:

- Observaciones muy parecidas pertenecen a un mismo sub-grupo.
- Observaciones muy diferentes pertenecen a distintos sub-grupos.

## Agrupamiento de datos (Clustering)

Conjunto de métodos **no-supervisados** para dividir un grupo de datos en sub-grupos de tal forma que:

- Observaciones muy parecidas pertenecen a un mismo sub-grupo.
- Observaciones muy diferentes pertenecen a distintos sub-grupos.

Algunos de los métodos más conocidos:

- $k$ -means
- $k$ -medoids
- Hierarchical clustering

## Agrupamiento de datos (Clustering)

Conjunto de métodos **no-supervisados** para dividir un grupo de datos en sub-grupos de tal forma que:

- Observaciones muy parecidas pertenecen a un mismo sub-grupo.
- Observaciones muy diferentes pertenecen a distintos sub-grupos.

Algunos de los métodos más conocidos:

- *k-means*
- *k-medoids*
- Hierarchical clustering

## *k*-means

---

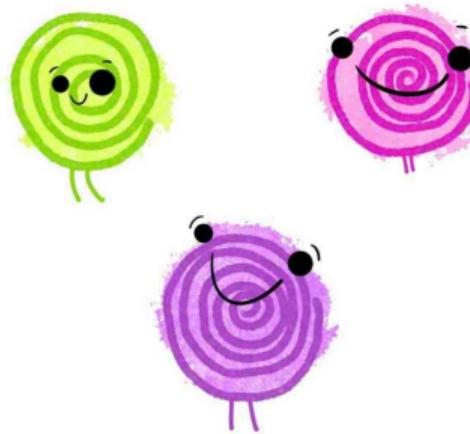
# k-means clustering

OBSERVATIONS



- assign each observation to one of k clusters based on the nearest cluster centroid.

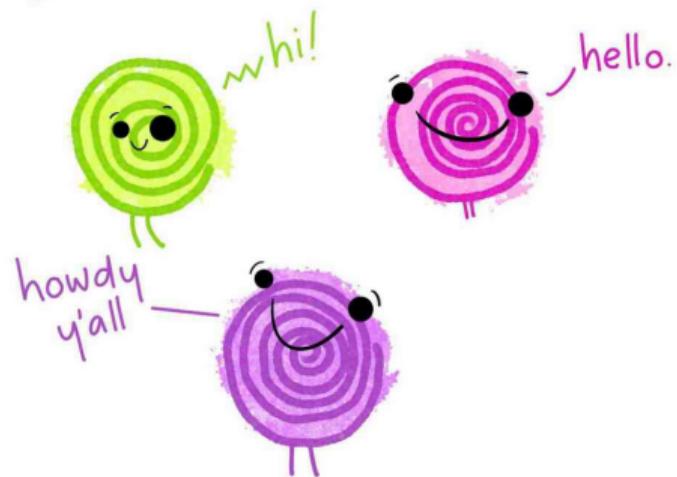
cluster CENTROIDS



①

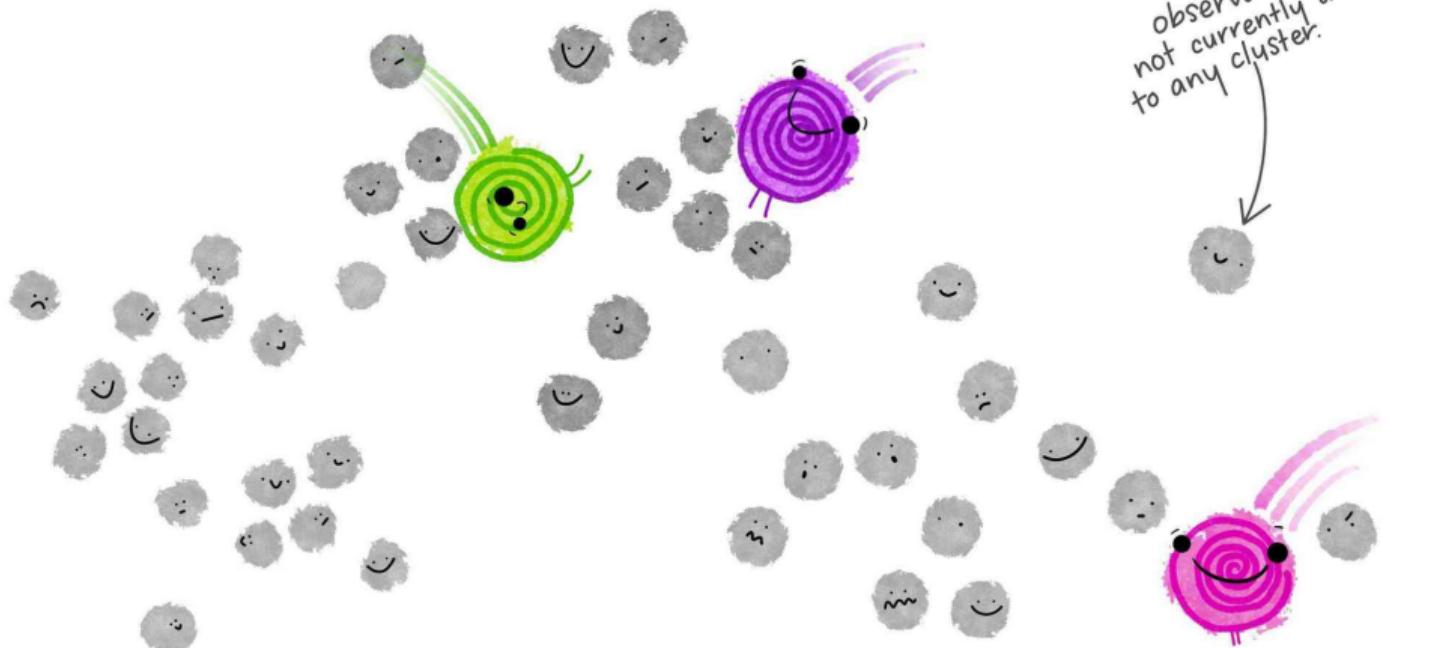
Specify the number of clusters (in this example,  $k=3$ ).

Then imagine  $k$  cluster centroids are created.



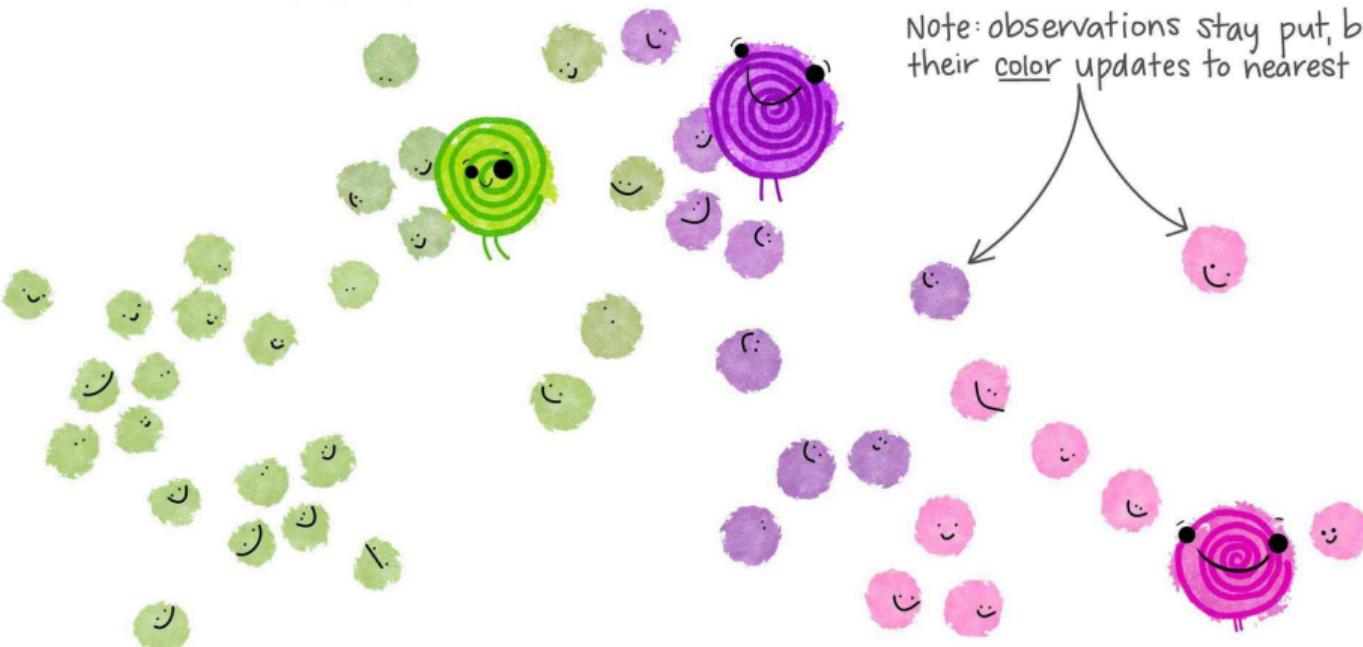
②

Those k centroids get randomly placed in your space.



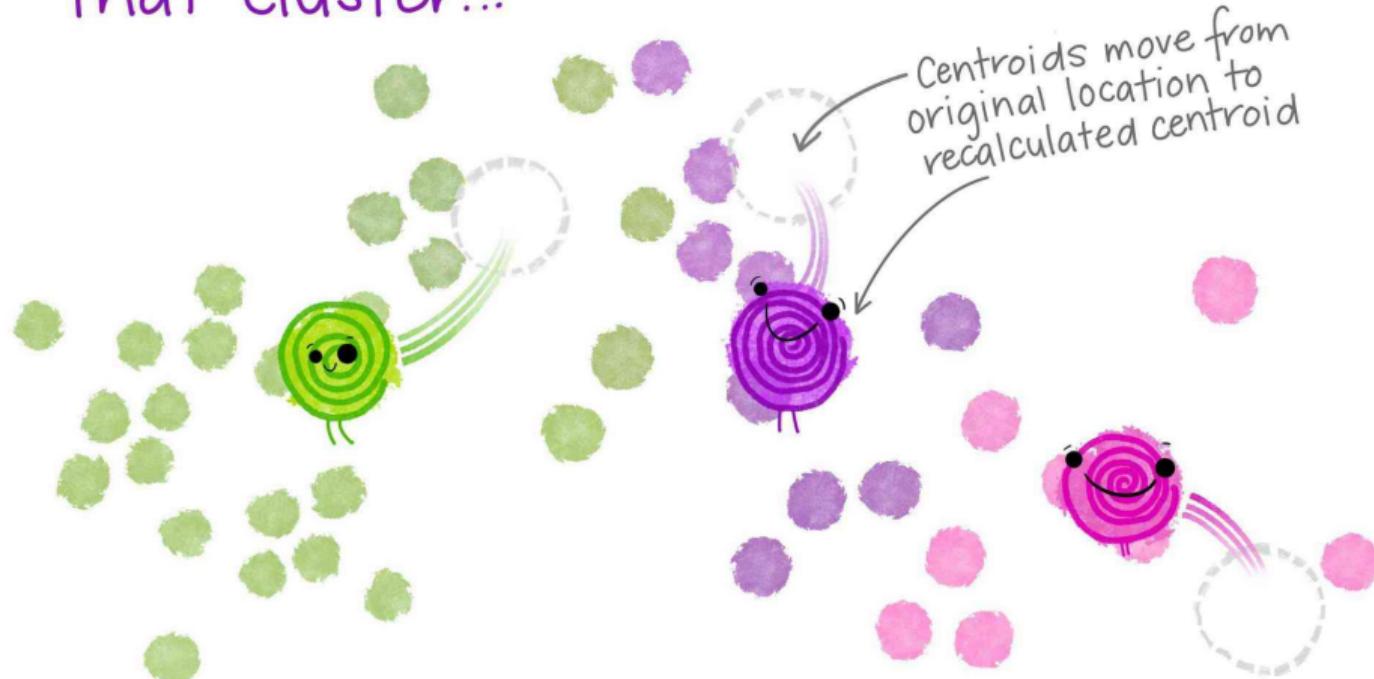
③

Each observation gets temporarily "assigned" to its closest centroid.  
(e.g. by Euclidean distance)



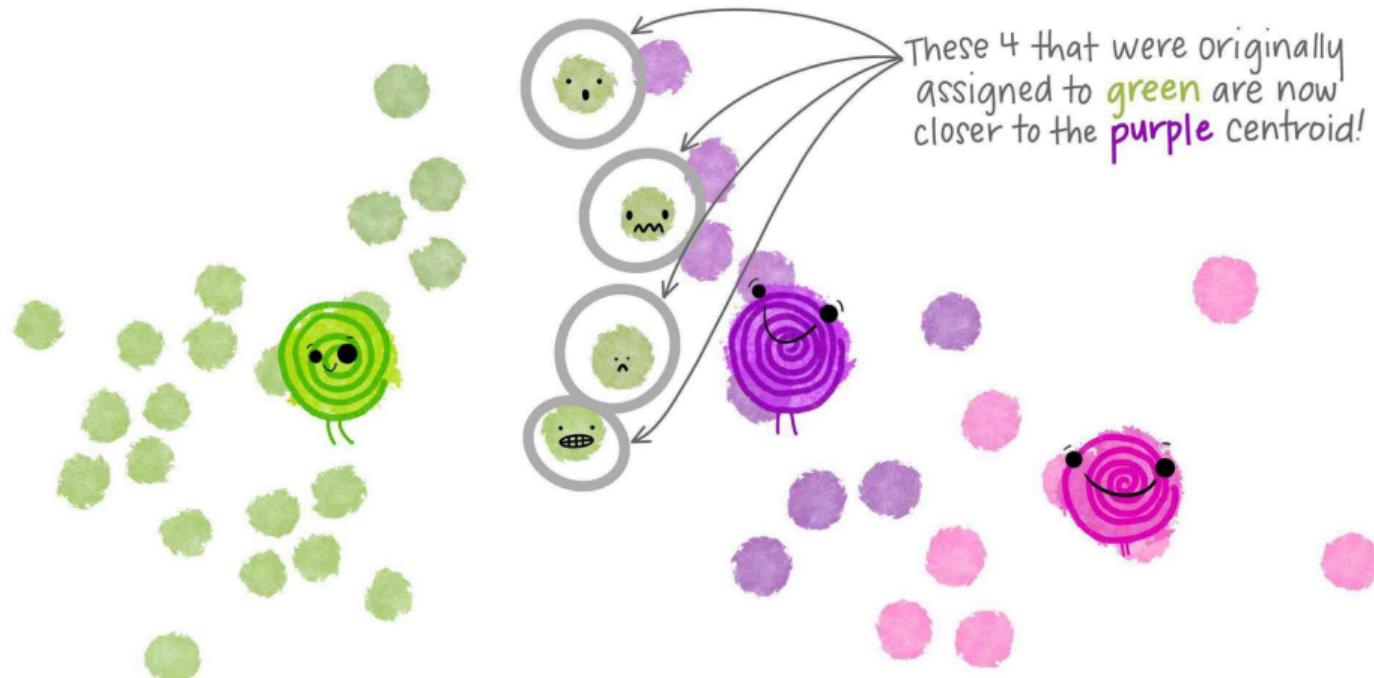
4

Then the centroid of each cluster is calculated based on all observations assigned to that cluster...





UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!

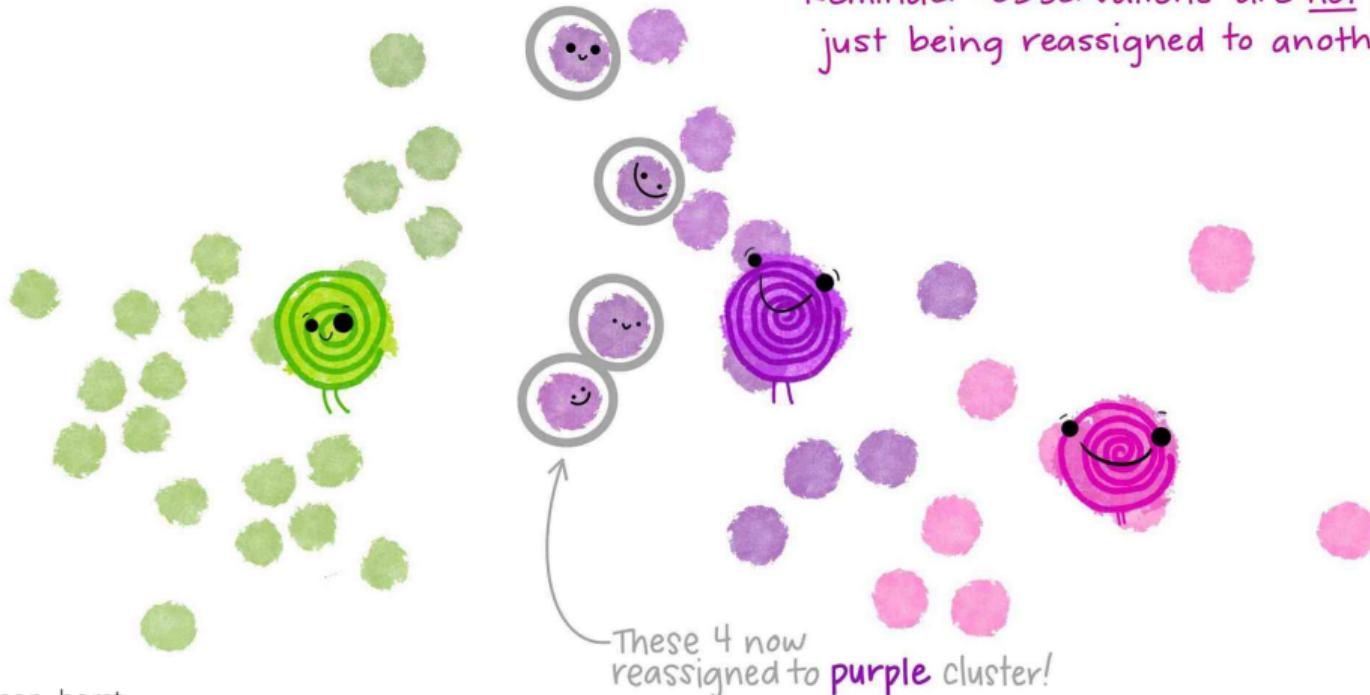




## NO PROBLEM!

Observations get reassigned\* to a different cluster based on the recalculated centroid.

\*Reminder: observations are not moving, just being reassigned to another cluster.



6

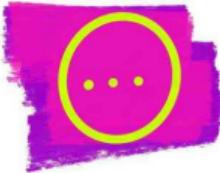
But now that observations have been reassigned,  
the centroids need to move again [recalculate  
centroids from updated clusters]



7

Again, now observations are reassigned as needed to the closest centroid.





Then the centroid for each cluster  
is recalculated...



...which means observations will be reassigned...



# That iterative process of

Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

↳ Reassign observations to nearest centroid



Continues until nothing is moving  
or being reassigned anymore!

fin

Which means the iteration is done and each observation is assigned to its final cluster.



## Ejemplo $k$ -means con $k = 2$

En la librería `stats`, paquete base de R, tenemos la función `kmeans()`

```
k <- 2
k2res <- kmeans(B, k)
str(k2res)
```

List of 9

- \$ cluster : int [1:680] 1 1 1 1 1 1 1 1 1 1 ...
- \$ centers : num [1:2, 1:6] 0.3371 1.1447 0.1176 0.5449 0.0385 ...
- ... attr(\*, "dimnames")=List of 2
- ... ..\$ : chr [1:2] "1" "2"
- ... ..\$ : chr [1:6] "MEMORY" "LANGUAGE" "EXECUTIVE" "VISUOSPATIAL" ...
- \$ totss : num 4872
- \$ withinss : num [1:2] 1930 1082
- \$ tot.withinss: num 3012
- \$ betweenss : num 1860
- \$ size : int [1:2] 543 137
- \$ iter : int 1
- \$ ifault : int 0
- attr(\*, "class")= chr "kmeans"

## Ejemplo $k$ -means con $k = 2$

En la librería `stats`, paquete base de R, tenemos la función `kmeans()`

```
k <- 2
k2res <- kmeans(B, k)
str(k2res)
```

List of 9  
\$ cluster : int [1:680] 1 1 1 1 1 1 1 1 1 1 ...  
\$ centers : num [1:2, 1:6] 0.3371 1.1447 0.1176 0.5449 0.0385 ...  
... attr(\*, "dimnames")=List of 2  
... ..\$ : chr [1:2] "1" "2"  
... ..\$ : chr [1:6] "MEMORY" "LANGUAGE" "EXECUTIVE" "VISUOSPATIAL" ...  
\$ totss : num 4872  
\$ withinss : num [1:2] 1930 1082  
\$ tot.withinss: num 3012  
\$ betweenss : num 1860  
\$ size : int [1:2] 543 137  
\$ iter : int 1  
\$ ifault : int 0  
- attr(\*, "class")= chr "kmeans"

Los grupos asignados a las 680 observaciones están en `k2res$cluster`

## Ejemplo $k$ -means con $k = 3$

```
k <- 3
k3res <- kmeans(B, k)
str(k3res)
```

List of 9  
\$ cluster : int [1:680] 3 3 3 2 3 3 3 3 3 3 ...  
\$ centers : num [1:3, 1:6] 1.154 0.767 0.183 0.544 0.583 ...  
... attr(\*, "dimnames")=List of 2  
... ..\$ : chr [1:3] "1" "2" "3"  
... ..\$ : chr [1:6] "MEMORY" "LANGUAGE" "EXECUTIVE" "VISUOSPATIAL" ...  
\$ totss : num 4872  
\$ withinss : num [1:3] 869 750 772  
\$ tot.withinss: num 2390  
\$ betweenss : num 2481  
\$ size : int [1:3] 117 174 389  
\$ iter : int 3  
\$ ifault : int 0  
- attr(\*, "class")= chr "kmeans"

## Ejemplo $k$ -means con $k = 3$

```
k <- 3
k3res <- kmeans(B, k)
str(k3res)
```

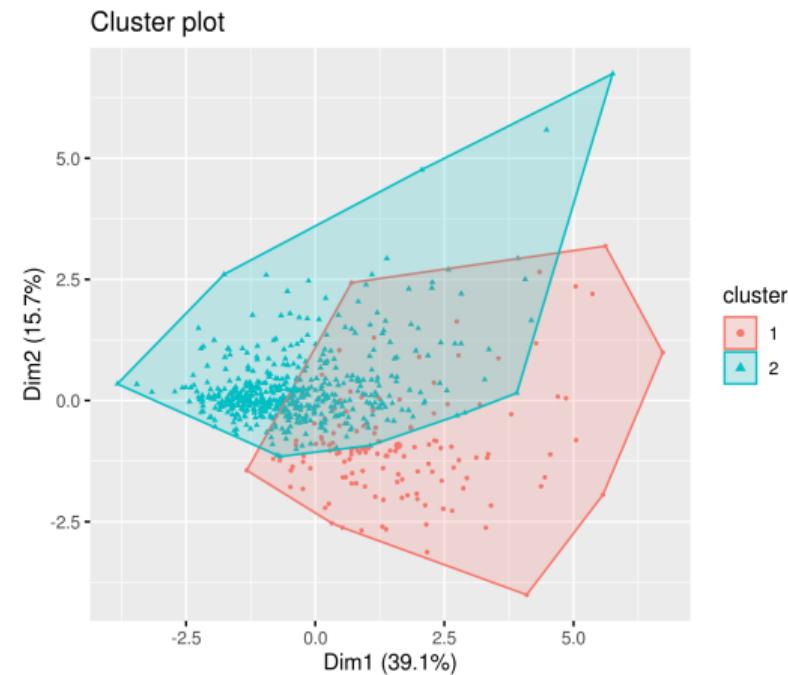
List of 9  
\$ cluster : int [1:680] 3 3 3 2 3 3 3 3 3 3 ...  
\$ centers : num [1:3, 1:6] 1.154 0.767 0.183 0.544 0.583 ...  
... attr(\*, "dimnames")=List of 2  
... ..\$ : chr [1:3] "1" "2" "3"  
... ..\$ : chr [1:6] "MEMORY" "LANGUAGE" "EXECUTIVE" "VISUOSPATIAL" ...  
\$ totss : num 4872  
\$ withinss : num [1:3] 869 750 772  
\$ tot.withinss: num 2390  
\$ betweenss : num 2481  
\$ size : int [1:3] 117 174 389  
\$ iter : int 3  
\$ ifault : int 0  
- attr(\*, "class")= chr "kmeans"

Podemos comparar las dos particiones con `table(k2res$cluster, k3res$cluster)`. Los 3 sub-grupos no son necesariamente particiones de los 2 primeros sub-grupos.

# ¿Cómo visualizar las particiones?

La función `fviz_cluster`, de la librería `factoextra`, grafica los datos en las 2 direcciones de mayor varianza\*:

```
fviz_cluster(k2res, B, geom = "point")
fviz_cluster(k3res, B, geom = "point")
k4res <- kmeans(B, 4)
fviz_cluster(k4res, B, geom = "point")
...
```

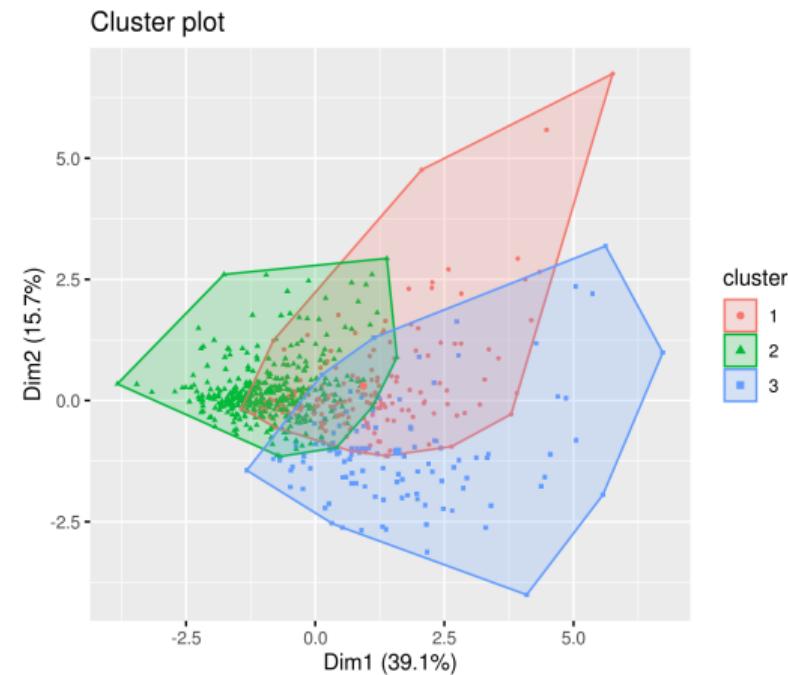


\* Usando análisis de componentes principales (PCA)

# ¿Cómo visualizar las particiones?

La función `fviz_cluster`, de la librería `factoextra`, grafica los datos en las 2 direcciones de mayor varianza\*:

```
fviz_cluster(k2res, B, geom = "point")
fviz_cluster(k3res, B, geom = "point")
k4res <- kmeans(B, 4)
fviz_cluster(k4res, B, geom = "point")
...
```

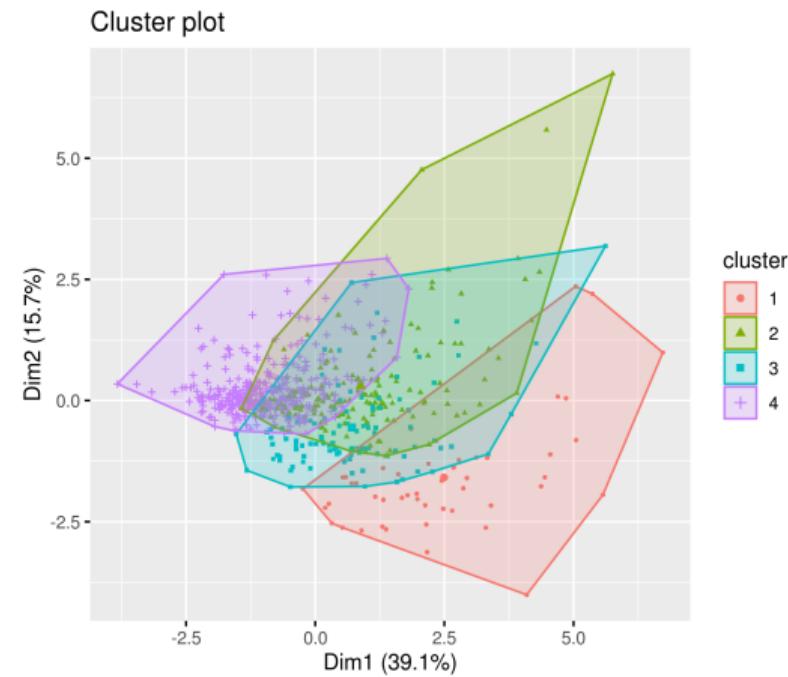


\* Usando análisis de componentes principales (PCA)

# ¿Cómo visualizar las particiones?

La función `fviz_cluster`, de la librería `factoextra`, grafica los datos en las 2 direcciones de mayor varianza\*:

```
fviz_cluster(k2res, B, geom = "point")
fviz_cluster(k3res, B, geom = "point")
k4res <- kmeans(B, 4)
fviz_cluster(k4res, B, geom = "point")
...
```

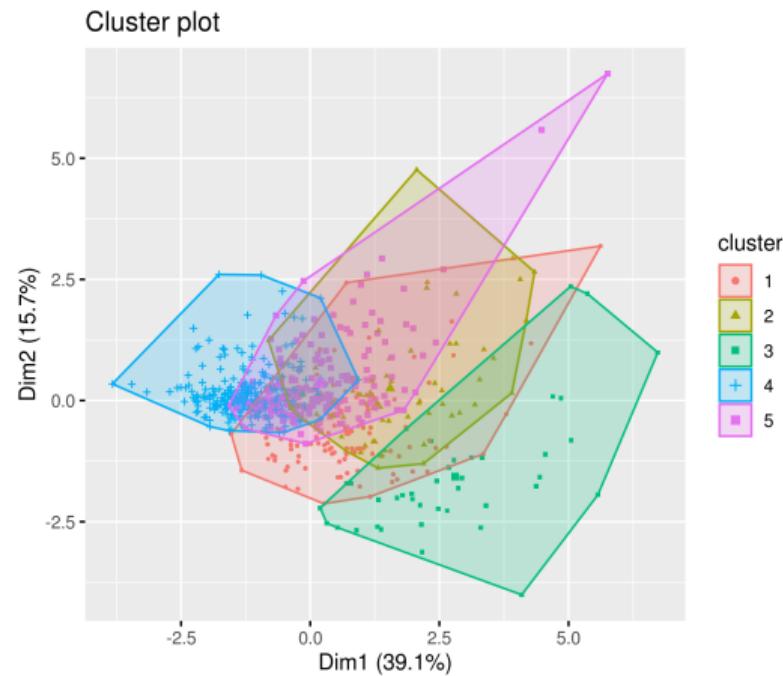


\* Usando análisis de componentes principales (PCA)

# ¿Cómo visualizar las particiones?

La función `fviz_cluster`, de la librería `factoextra`, grafica los datos en las 2 direcciones de mayor varianza\*:

```
fviz_cluster(k2res, B, geom = "point")
fviz_cluster(k3res, B, geom = "point")
k4res <- kmeans(B, 4)
fviz_cluster(k4res, B, geom = "point")
...
```



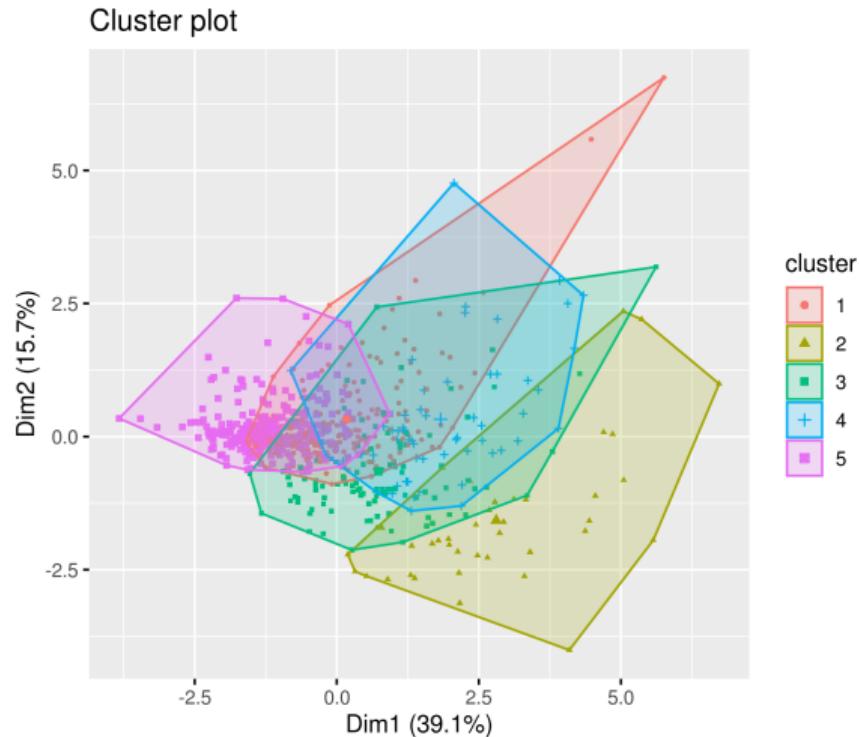
\* Usando análisis de componentes principales (PCA)

**Importante:** muchos (¿todos?) métodos de agrupamiento tienen inicialización aleatoria → el resultado no siempre es el mismo.

**Importante:** muchos (¿todos?) métodos de agrupamiento tienen inicialización aleatoria → el resultado no siempre es el mismo.

```
> k <- 5  
> set.seed(1987)  
> res1 <- kmeans(B, k)  
> set.seed(2020)  
> res2 <- kmeans(B, k)  
> table(res1$cluster, res2$cluster)
```

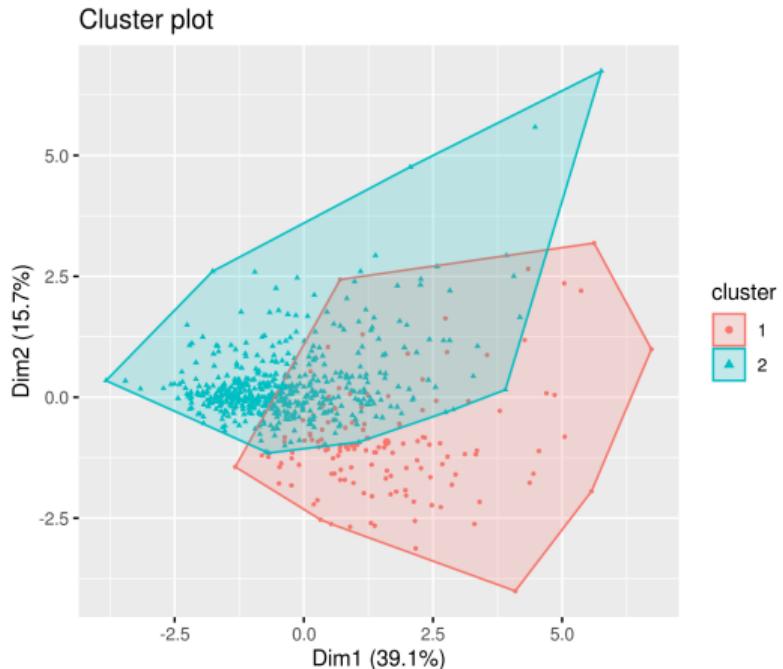
	1	2	3	4	5
1	29	0	0	0	263
2	116	0	1	2	19
3	52	0	2	55	0
4	0	1	100	0	0
5	0	39	0	1	0



¿Cómo elegir  $k$ ?

---

# ¿Qué pasa cuando aumenta el número de sub-grupos?

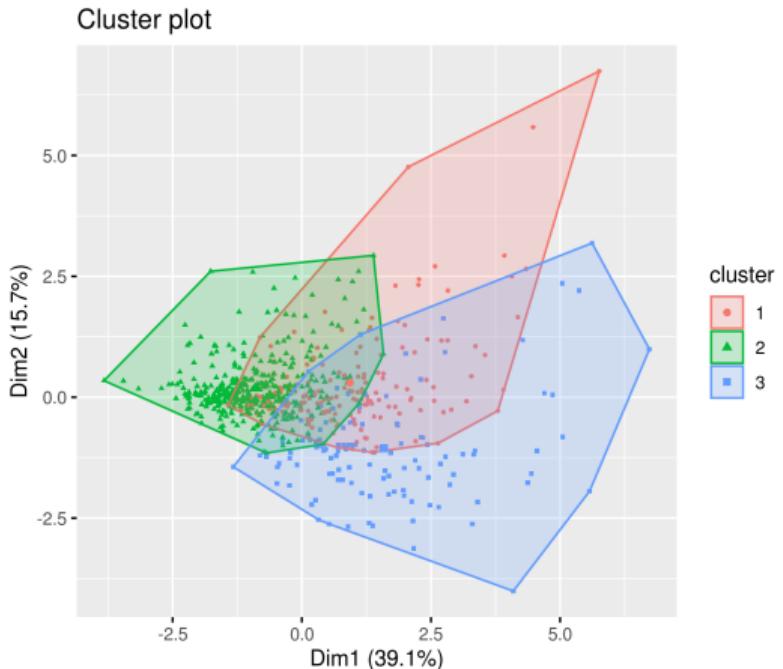


Suma de los cuadrados de las distancias al centro del sub-grupo (`$withinss`):

2. 1081.833 1930.495

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

# ¿Qué pasa cuando aumenta el número de sub-grupos?



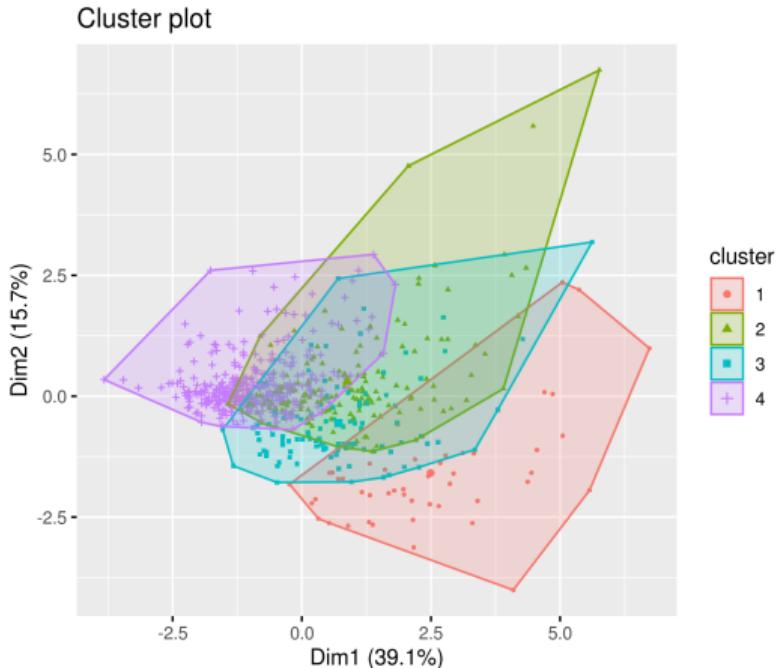
Suma de los cuadrados de las distancias al centro del sub-grupo (\$withinss):

2. 1081.833 1930.495

3. 759.5350 762.2628 868.6328

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

# ¿Qué pasa cuando aumenta el número de sub-grupos?

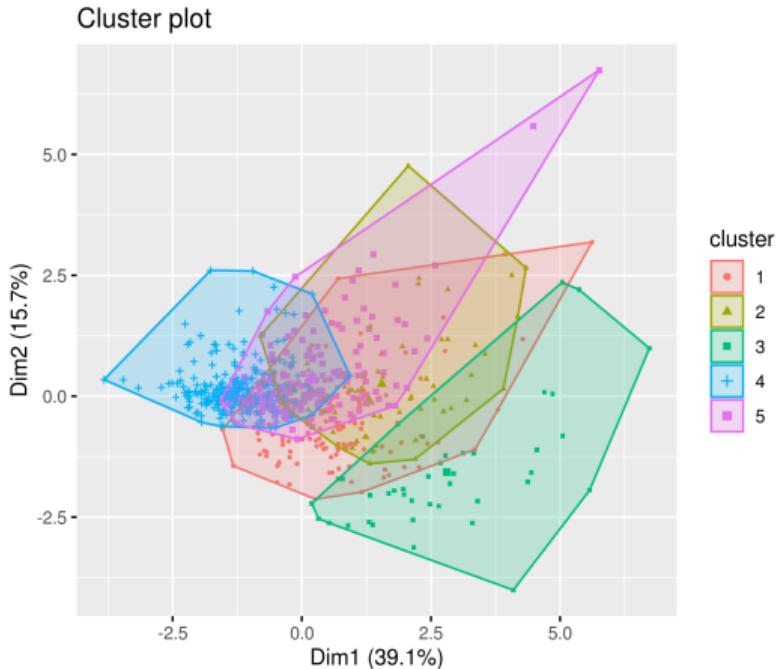


Suma de los cuadrados de las distancias al centro del sub-grupo (\$withinss):

2. 1081.833 1930.495
3. 759.5350 762.2628 868.6328
4. 347.6101 688.4191 390.6727 618.7741

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

# ¿Qué pasa cuando aumenta el número de sub-grupos?

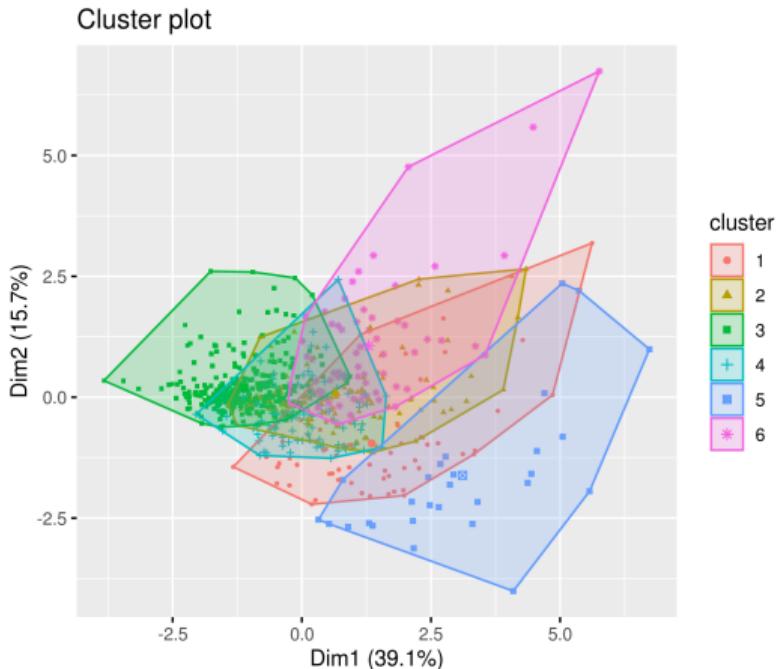


Suma de los cuadrados de las distancias al centro del sub-grupo (\$withinss):

2. 1081.833 1930.495
3. 759.5350 762.2628 868.6328
4. 347.6101 688.4191 390.6727 618.7741
5. 390.9384 309.5926 277.6631 368.1043  
514.1842

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

# ¿Qué pasa cuando aumenta el número de sub-grupos?

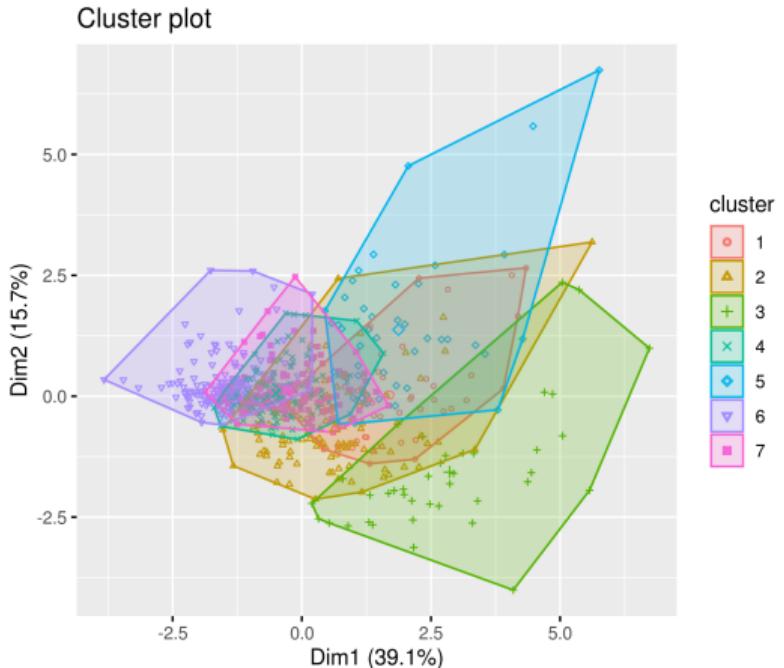


Suma de los cuadrados de las distancias al centro del sub-grupo (\$withinss):

2. 1081.833 1930.495
3. 759.5350 762.2628 868.6328
4. 347.6101 688.4191 390.6727 618.7741
5. 390.9384 309.5926 277.6631 368.1043  
514.1842
6. 300.4036 398.3652 424.8112 189.4664  
203.1081 247.0580

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

# ¿Qué pasa cuando aumenta el número de sub-grupos?



Suma de los cuadrados de las distancias al centro del sub-grupo (\$withinss):

2. 1081.833 1930.495
3. 759.5350 762.2628 868.6328
4. 347.6101 688.4191 390.6727 618.7741
5. 390.9384 309.5926 277.6631 368.1043  
514.1842
6. 300.4036 398.3652 424.8112 189.4664  
203.1081 247.0580
7. 213.9738 358.3246 270.7691 173.4431  
218.0087 174.3776 199.9581

\* En este caso, k-means con distancia euclídea, esta es la varianza intra-grupo

$$\text{Variabilidad total} \geq \sum_{\text{sub-grupos}} \text{Variabilidad intra-grupo}$$

$$\sum_{\text{observaciones}} (\text{distancia al centro del grupo})^2 \geq \sum_{\text{sub-grupos}} \sum_{\substack{\text{observaciones} \\ \text{en sub-grupo}}} (\text{distancia al centro del sub-grupo})^2$$

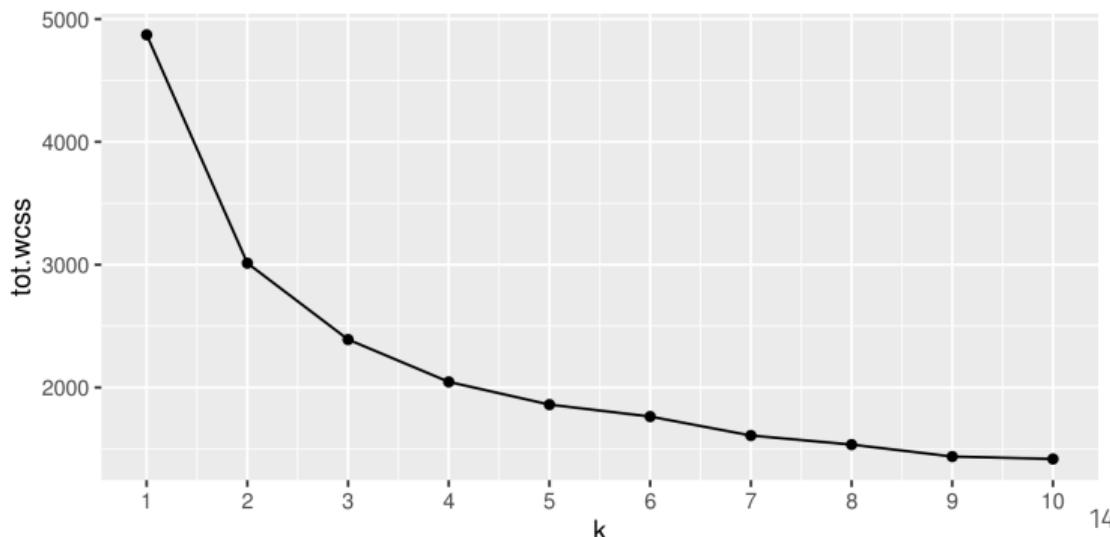
$$\$totss \geq \$tot.withinss$$

Variabilidad total  $\geq \sum_{\text{sub-grupos}} \text{Variabilidad intra-grupo}$

$\sum_{\text{observaciones}} (\text{distancia al centro del grupo})^2 \geq \sum_{\text{sub-grupos}} \sum_{\substack{\text{observaciones} \\ \text{en sub-grupo}}} (\text{distancia al centro del sub-grupo})^2$

$$\$totss \geq \$tot.withinss$$

## Método del codo (Elbow method)



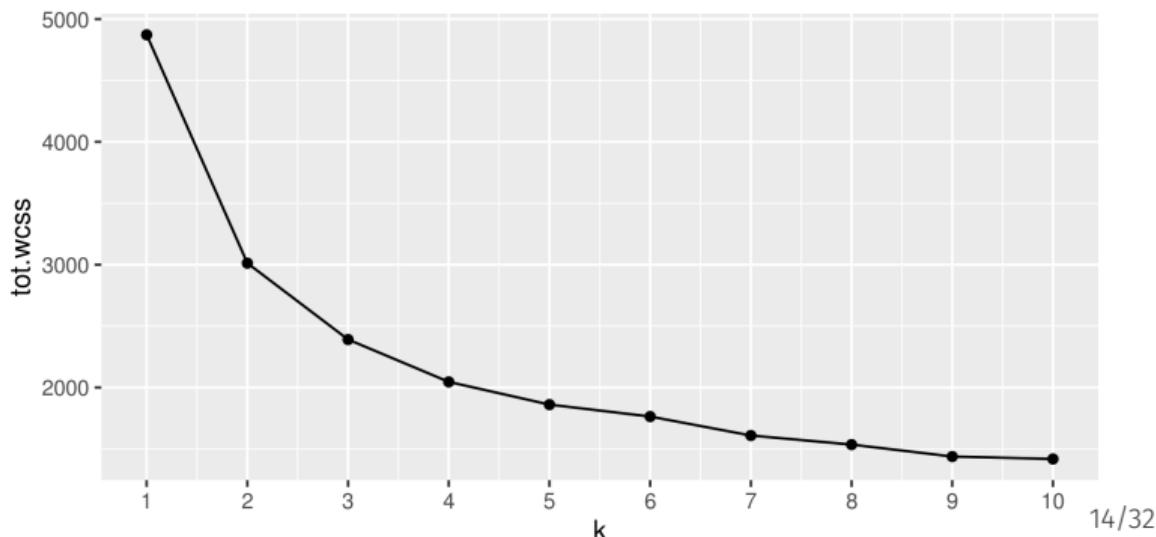
Variabilidad total  $\geq \sum_{\text{sub-grupos}} \text{Variabilidad intra-grupo}$

$\sum_{\text{observaciones}} (\text{distancia al centro del grupo})^2 \geq \sum_{\text{sub-grupos}} \sum_{\substack{\text{observaciones} \\ \text{en sub-grupo}}} (\text{distancia al centro del sub-grupo})^2$

$$\$totss \geq \$tot.withinss$$

## Método del codo (Elbow method)

meh!

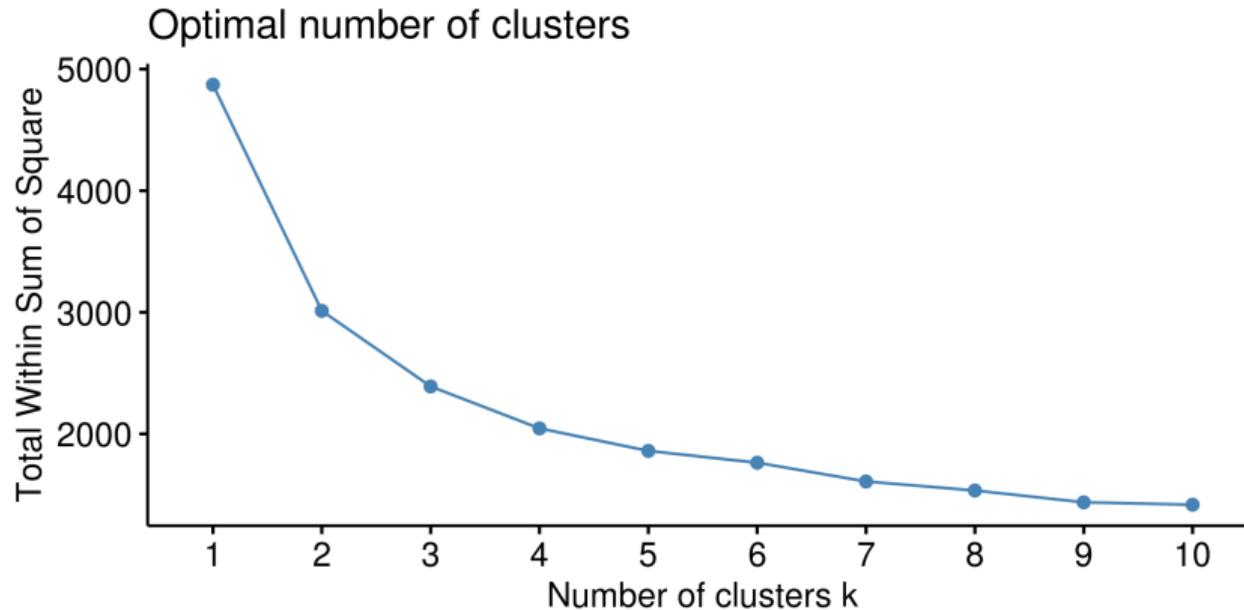


Esta gráfica también la podemos generar con la función `fviz_nbclust`:

```
fviz_nbclust(B, kmeans, method = "wss")
```

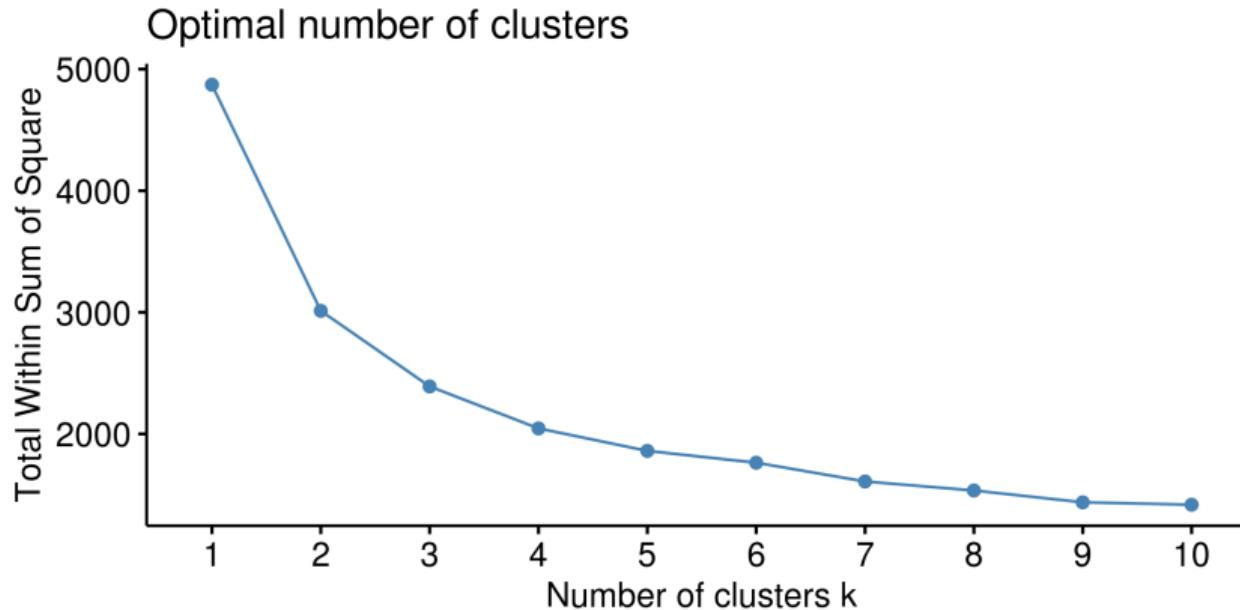
Esta gráfica también la podemos generar con la función `fviz_nbclust`:

```
fviz_nbclust(B, kmeans, method = "wss")
```



Esta gráfica también la podemos generar con la función `fviz_nbclust`:

```
fviz_nbclust(B, kmeans, method = "wss")
```



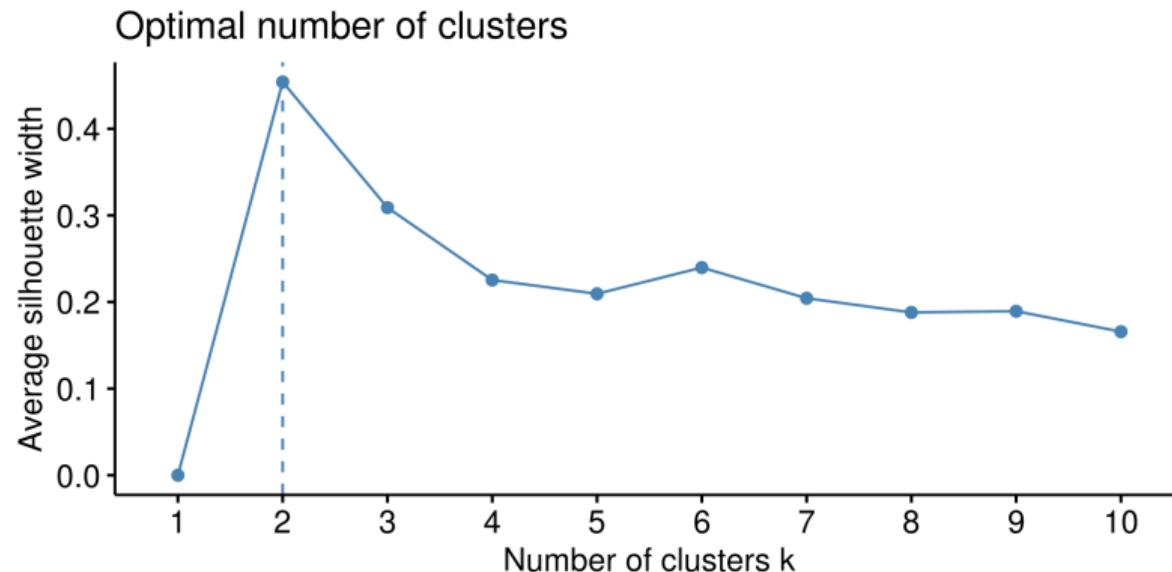
¿Dónde se ve un codo?

**Método de la silueta:** basado en que tan lejos (o cerca) están las observaciones a los sub-grupos vecinos.

```
fviz_nbclust(B, kmeans, method = "silhouette")
```

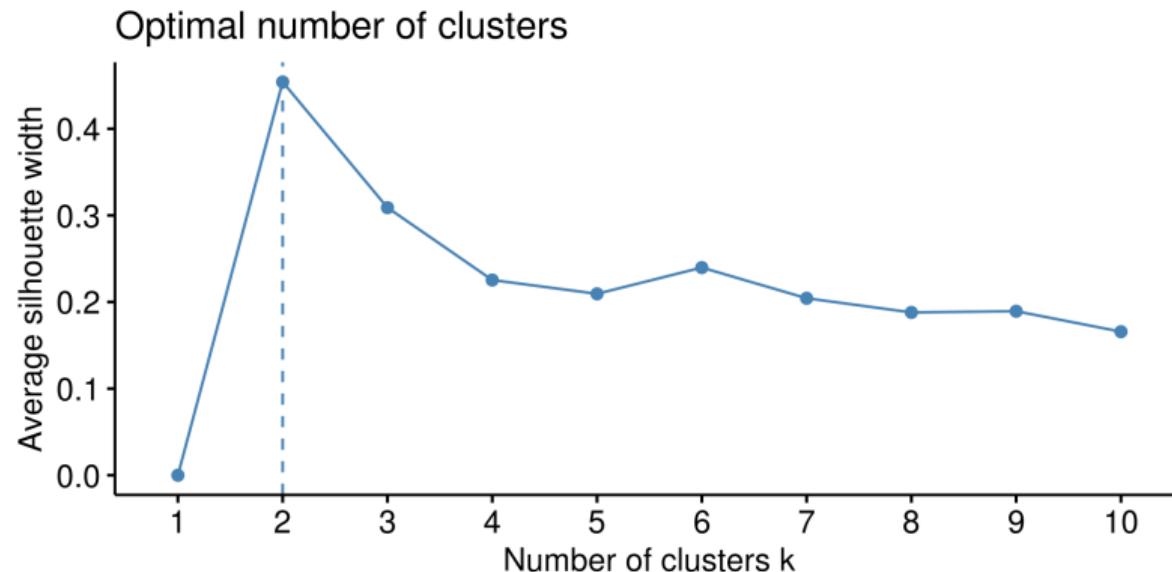
**Método de la silueta:** basado en que tan lejos (o cerca) están las observaciones a los sub-grupos vecinos.

```
fviz_nbclust(B, kmeans, method = "silhouette")
```



**Método de la silueta:** basado en que tan lejos (o cerca) están las observaciones a los sub-grupos vecinos.

```
fviz_nbclust(B, kmeans, method = "silhouette")
```



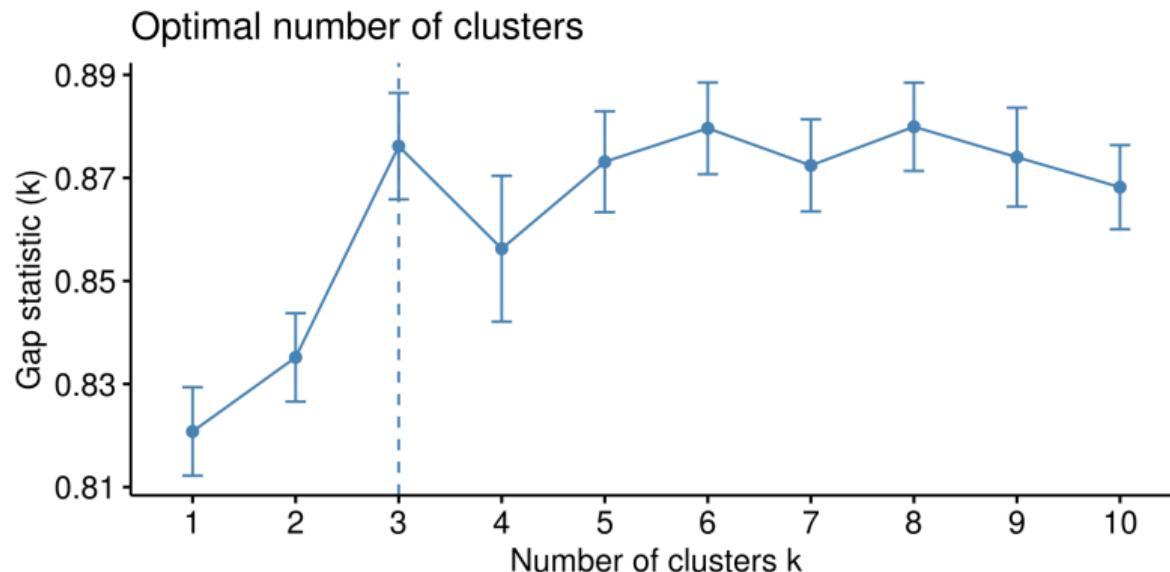
Número sugerido de sub-grupos: 2

**Método del *gap statistic*:** Compara las varianzas intra-grupo observadas Vs. varianzas intra-grupo simuladas.

```
fviz_nbclust(B, kmeans, method = "gap_stat")
```

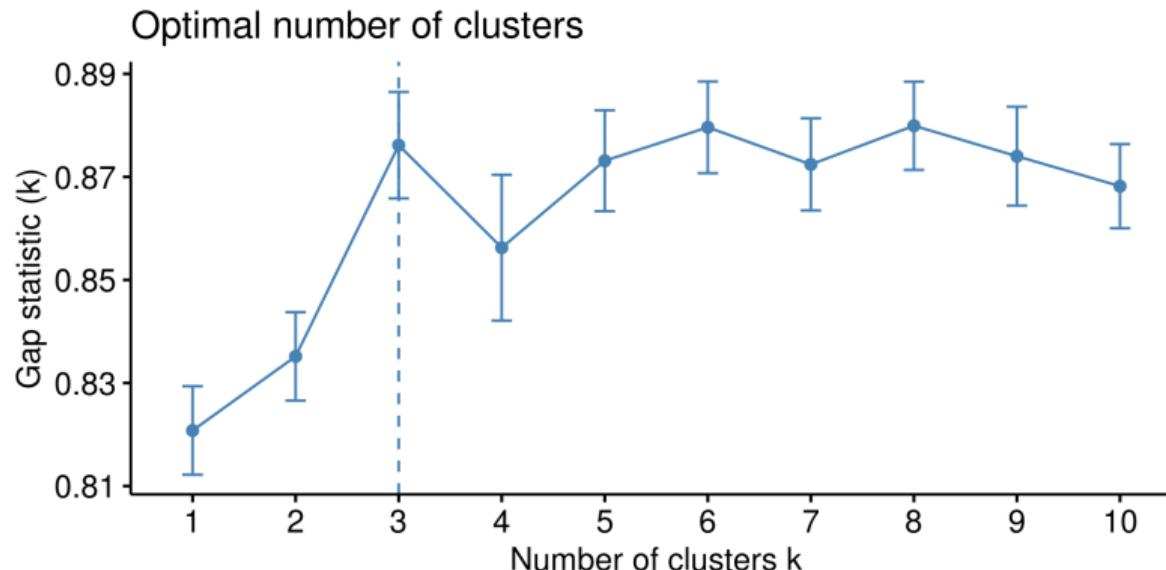
**Método del *gap statistic*:** Compara las varianzas intra-grupo observadas Vs. varianzas intra-grupo simuladas.

```
fviz_nbclust(B, kmeans, method = "gap_stat")
```



**Método del *gap statistic*:** Compara las varianzas intra-grupo observadas Vs. varianzas intra-grupo simuladas.

```
fviz_nbclust(B, kmeans, method = "gap_stat")
```



Número sugerido de sub-grupos: 3

Hay **± TREINTA** índices/criterios diferentes:

```
NbClust(data = B, method = "kmeans")
```

Hay **± TREINTA** índices/criterios diferentes:

```
NbClust(data = B, method = "kmeans")
```

```
*****
```

```
* Among all indices:
```

- \* 7 proposed 2 as the best number of clusters
- \* 11 proposed 3 as the best number of clusters
- \* 2 proposed 5 as the best number of clusters
- \* 1 proposed 10 as the best number of clusters
- \* 1 proposed 14 as the best number of clusters
- \* 2 proposed 15 as the best number of clusters

```
***** Conclusion *****
```

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

```
*****
```

Hay **± TREINTA** índices/criterios diferentes:

```
NbClust(data = B, method = "kmeans")
```

```
*****
```

```
* Among all indices:
```

- \* 7 proposed 2 as the best number of clusters
- \* 11 proposed 3 as the best number of clusters
- \* 2 proposed 5 as the best number of clusters
- \* 1 proposed 10 as the best number of clusters
- \* 1 proposed 14 as the best number of clusters
- \* 2 proposed 15 as the best number of clusters

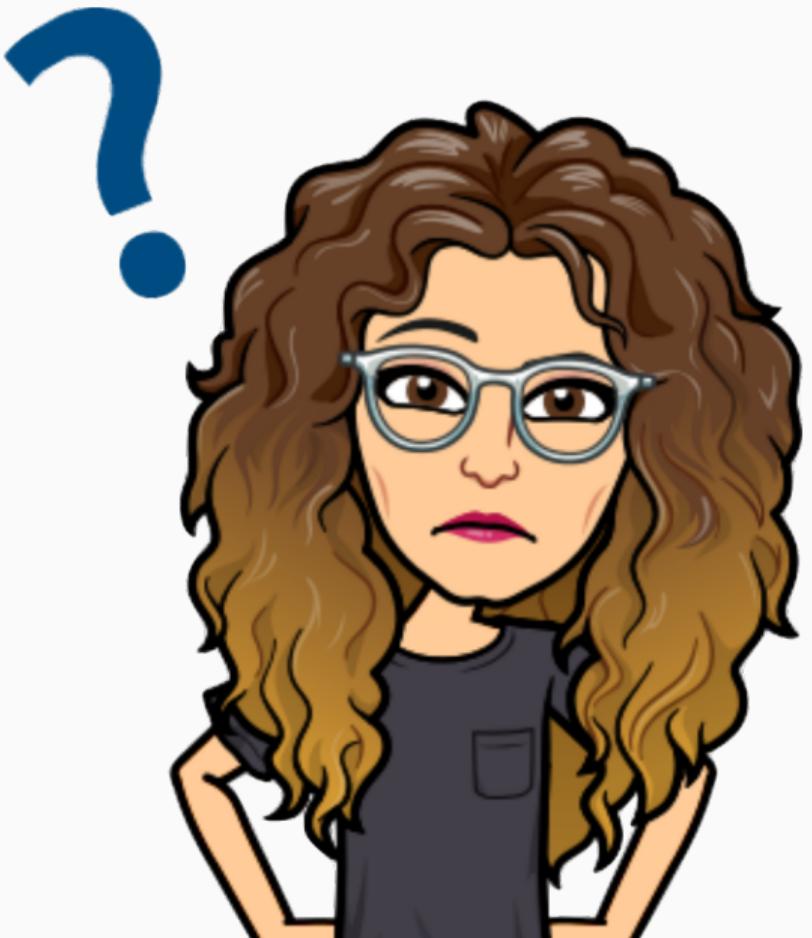
```
***** Conclusion *****
```

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

```
*****
```

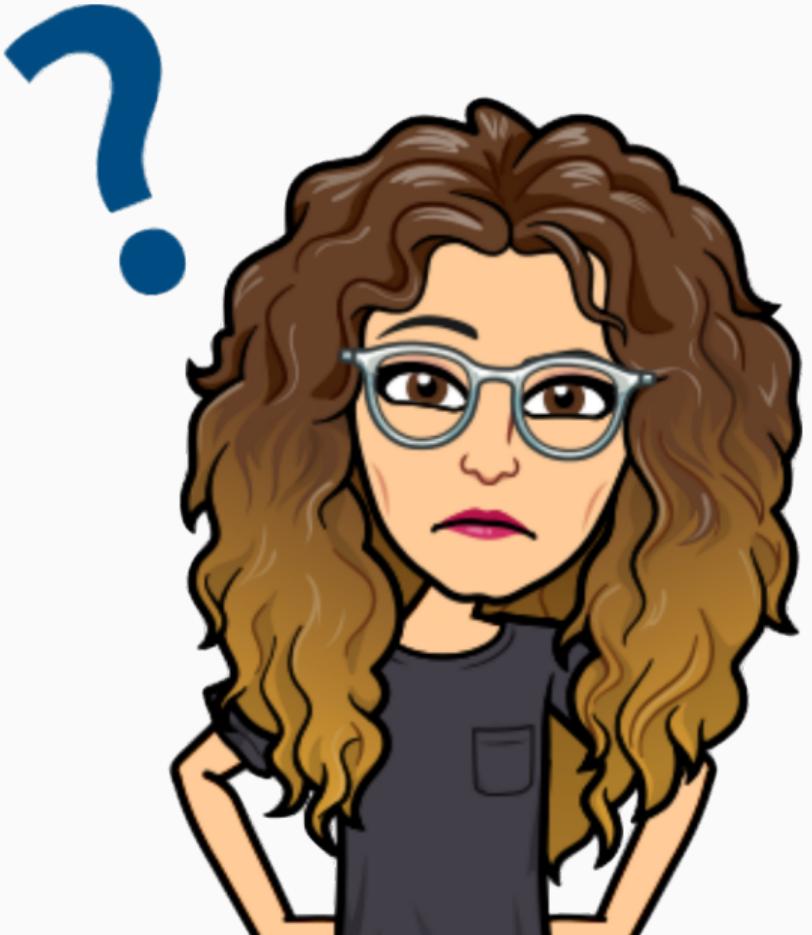
Número de sub-grupos sugerido por la mayoría: 3

Ajá,  
y entonces?



Ajá,  
y entonces?

Usar el criterio propio para tomar  
una *decisión informada*

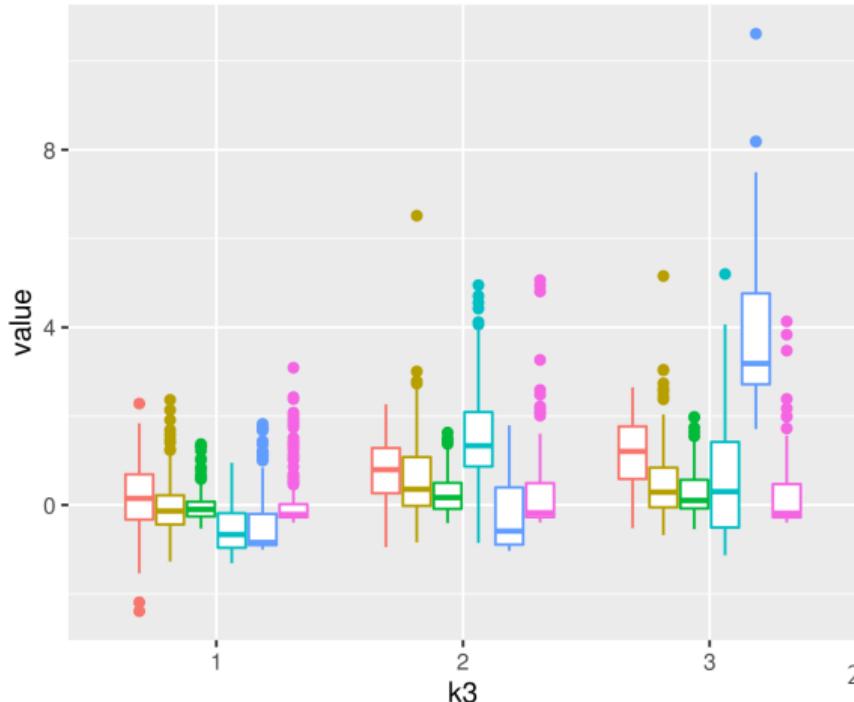


## Resultado con $k = 3$

```
res <- kmeans(B, 3)
Amci$k3 <- as.factor(res$cluster)
nd <- colnames(B)
m <- melt(Amci, measure.vars = nd)
ggplot(m, aes(k3, value)) +
  geom_boxplot(aes(colour = variable)) +
  theme(legend.position = "top")
```

## Resultado con $k = 3$

```
res <- kmeans(B, 3)
Amci$k3 <- as.factor(res$cluster)
nd <- colnames(B)
m <- melt(Amci, measure.vars = nd)
ggplot(m, aes(k3, value)) +
  geom_boxplot(aes(colour = variable)) +
  theme(legend.position = "top")
```



## Hasta ahora...

Hemos usado ***k-means*** y hemos asumido que la distancia entre observaciones es la **distancia euclídea** (por defecto).

## Hasta ahora...

Hemos usado ***k-means*** y hemos asumido que la distancia entre observaciones es la **distancia euclídea** (por defecto).

Sobre ***k-means***:

- Como la media, es sensible a *outliers*
- Los centros de cada sub-grupo no están necesariamente en el conjunto de observaciones.
- Es necesario conocer el espacio donde están las observaciones.

## Hasta ahora...

Hemos usado ***k-means*** y hemos asumido que la distancia entre observaciones es la **distancia euclídea** (por defecto).

### Sobre *k-means*:

- Como la media, es sensible a *outliers*
- Los centros de cada sub-grupo no están necesariamente en el conjunto de observaciones.
- Es necesario conocer el espacio donde están las observaciones.

### Sobre la distancia euclídea:

- Se asume ortogonalidad entre dimensiones ( $\rightarrow$  independencia?)
- Todas las dimensiones tienen la misma importancia.
- Cosas raras empiezan a pasar en espacios de muchas dimensiones.

## Distancias entre observaciones

---

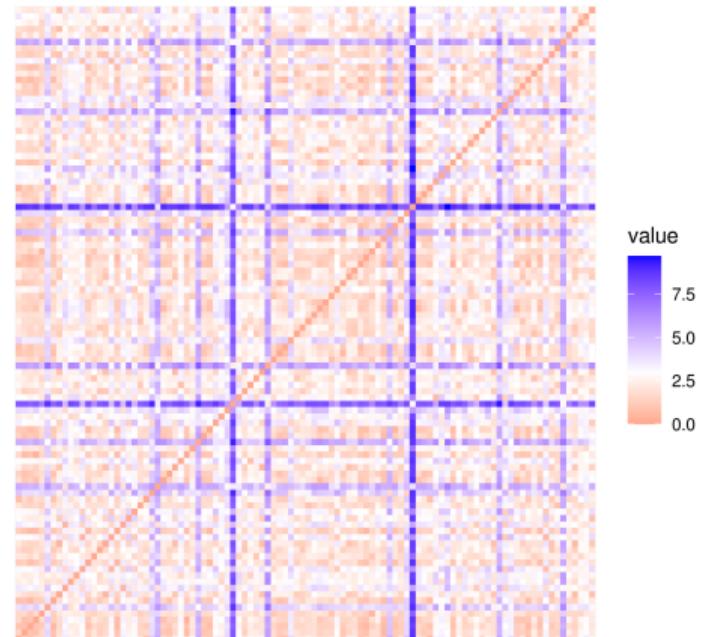
Las funciones `dist` y `get_dist` permiten calcular diferentes distancias entre observaciones.

```
fviz_dist(d, show_labels = FALSE)
```

```
tmp_B <- B[sample(1:nrow(B), 100),]
```

Euclideana ( $L^2$ ):

```
dist(tmp_B, method = "euclidean")
```



Las funciones `dist` y `get_dist` permiten calcular diferentes distancias entre observaciones.

```
fviz_dist(d, show_labels = FALSE)
```

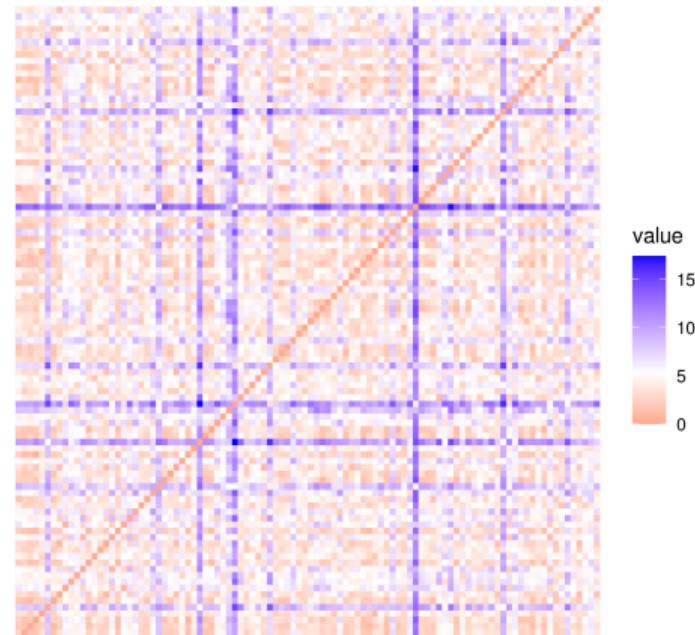
```
tmp_B <- B[sample(1:nrow(B), 100),]
```

Euclideana ( $L^2$ ):

```
dist(tmp_B, method = "euclidean")
```

Manhattan ( $L^1$ ):

```
dist(tmp_B, method = "manhattan")
```



Las funciones `dist` y `get_dist` permiten calcular diferentes distancias entre observaciones.

```
fviz_dist(d, show_labels = FALSE)
```

```
tmp_B <- B[sample(1:nrow(B), 100),]
```

Euclideana ( $L^2$ ):

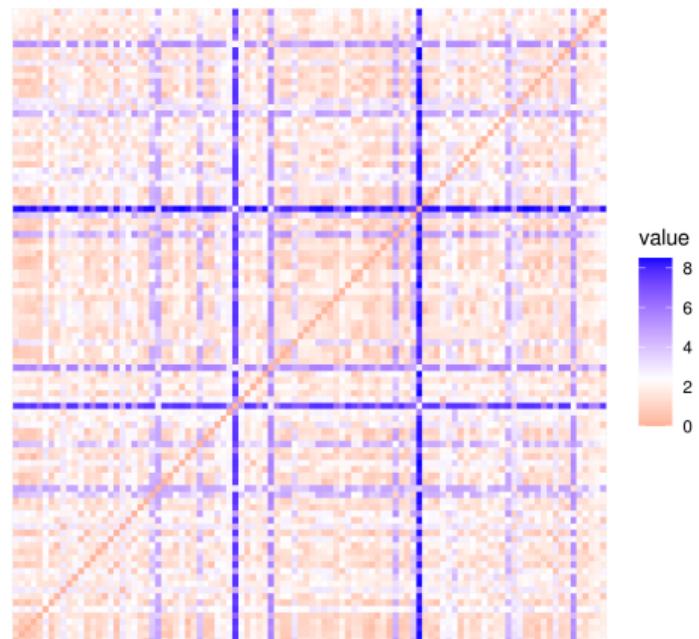
```
dist(tmp_B, method = "euclidean")
```

Manhattan ( $L^1$ ):

```
dist(tmp_B, method = "manhattan")
```

Maximo ( $L^\infty$ ):

```
dist(tmp_B, method = "maximum")
```



Las funciones `dist` y `get_dist` permiten calcular diferentes distancias entre observaciones.

```
fviz_dist(d, show_labels = FALSE)
```

```
tmp_B <- B[sample(1:nrow(B), 100),]
```

Euclideana ( $L^2$ ):

```
dist(tmp_B, method = "euclidean")
```

Manhattan ( $L^1$ ):

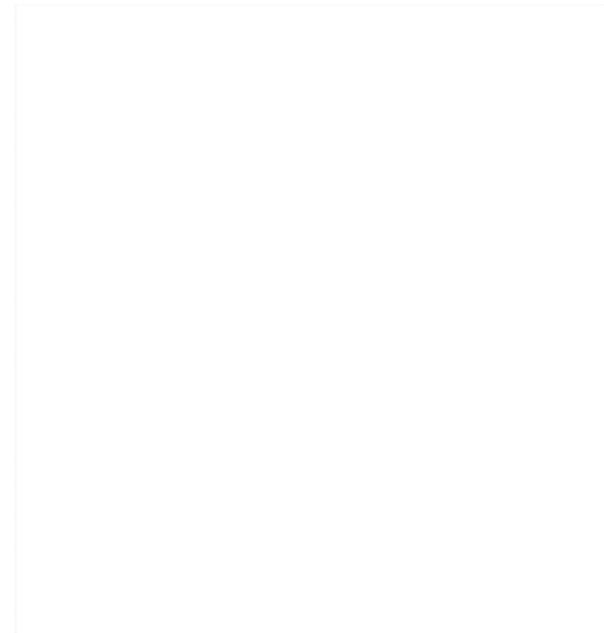
```
dist(tmp_B, method = "manhattan")
```

Maximo ( $L^\infty$ ):

```
dist(tmp_B, method = "maximum")
```

Binaria ( $L^0$ ):

```
dist(tmp_B, method = "binary")
```



value  
0

## De regreso a nuestros datos...

Tenemos los puntajes en 6 dominios cognitivos: memoria, lenguaje, función ejecutiva, habilidades visuoespaciales, orientación y atención.

## De regreso a nuestros datos...

Tenemos los puntajes en 6 dominios cognitivos: memoria, lenguaje, función ejecutiva, habilidades visuoespaciales, orientación y atención.

Estas variables **NO** son independientes!

## De regreso a nuestros datos...

Tenemos los puntajes en 6 dominios cognitivos: memoria, lenguaje, función ejecutiva, habilidades visuoespaciales, orientación y atención.

Estas variables **NO** son independientes!

Distancia de Mahalanobis: tiene en cuenta las correlaciones entre variables, reduce la redundancia.

$$d_M(i, j)^2 = (x_i - x_j)^T C^{-1} (x_i - x_j)$$

C: matriz de covarianza.

## De regreso a nuestros datos...

Tenemos los puntajes en 6 dominios cognitivos: memoria, lenguaje, función ejecutiva, habilidades visuoespaciales, orientación y atención.

Estas variables **NO** son independientes!

**Distancia de Mahalanobis:** tiene en cuenta las correlaciones entre variables, reduce la redundancia.

$$d_M(i, j)^2 = (x_i - x_j)^T C^{-1} (x_i - x_j)$$

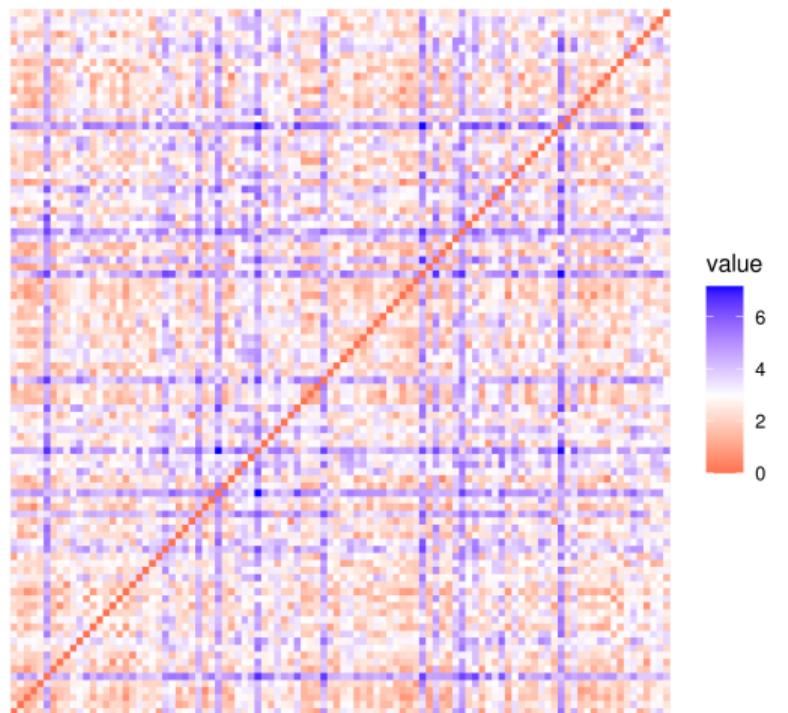
C: matriz de covarianza.

En R, la función `mahalanobis()` calcula la distancia de uno o varios puntos al centro de una distribución, no calcula la distancia entre pares de puntos.

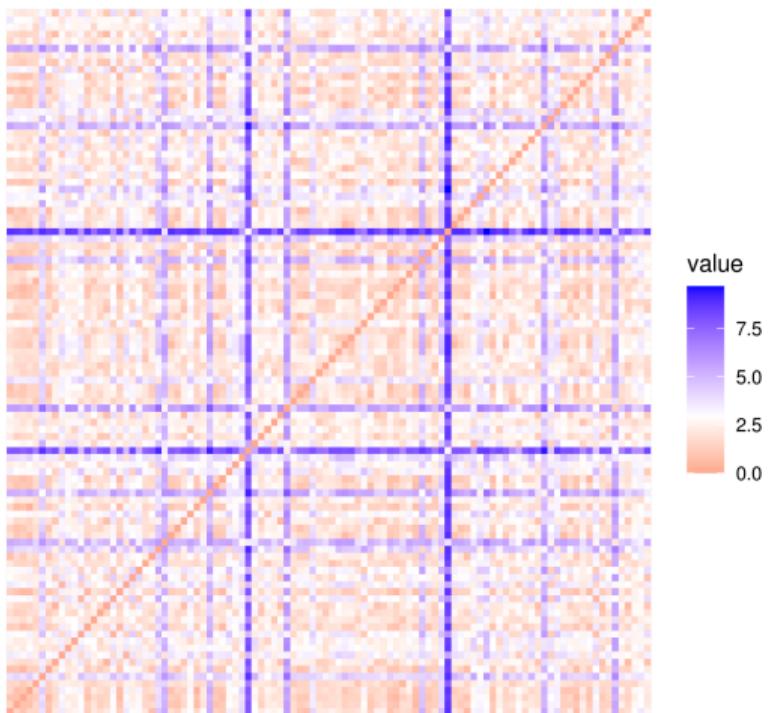
## Distancia de Mahalanobis hecha a mano

```
my_mah_dist <- function(M, invcov){  
  n <- nrow(M)  
  d_mah <- matrix(data = NA, nrow = n, ncol = n)  
  for (i in 1:n){  
    for (j in 1:i){  
      x <- as.matrix(M[i,] - M[j,])  
      d_mah[i,j] <- x %*% invcov %*% t(x)  
    }  
  }  
  d <- as.dist(sqrt(d_mah))  
  return(d)  
}  
  
d <- my_mah_dist(tmp_B, solve(cov(B)))
```

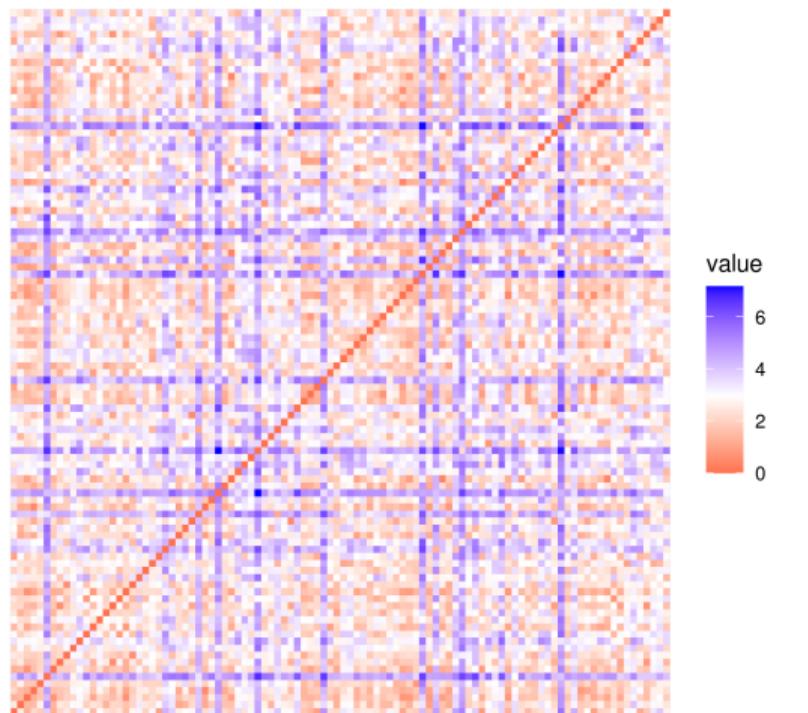
Distancia de Mahalanobis



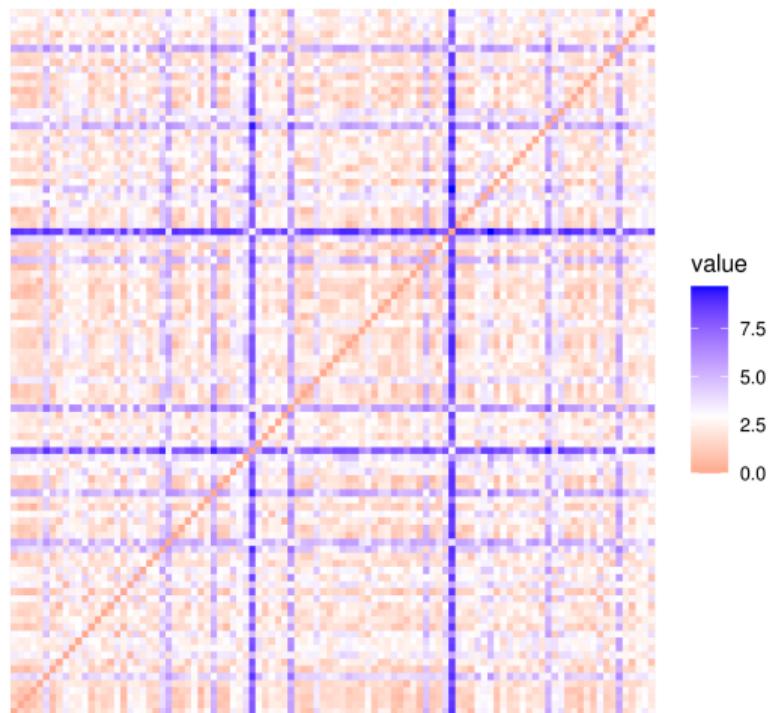
Distancia euclideana



Distancia de Mahalanobis



Distancia euclideana



El agrupamiento de datos depende completamente de la distancia!

## *k*-medoids

---

Necesitamos un método de agrupamiento que acepte nuestra propia distancia:  
*k-medoids*

Necesitamos un método de agrupamiento que acepte nuestra propia distancia:  
*k-medoids*

La función `pam()` de la librería `cluster` es una implementación de *k-medoids* o *Partition Around Medoids (PAM)*

Necesitamos un método de agrupamiento que acepte nuestra propia distancia:  
*k-medoids*

La función `pam()` de la librería `cluster` es una implementación de *k-medoids* o *Partition Around Medoids (PAM)*

```
> d <- my_mah_dist(B, solve(cov(B)))
> k <- 3
> pamres <- pam(d, k, diss = TRUE, pamonce = 5)           res$cluster:
> table(pamres$clustering, res$cluster)
```

	1	2	3
1	160	6	48
2	203	26	27
3	24	144	42

res\$cluster:  
partición con  
*k-means* y distancia  
euclideana.

Revisemos los criterios para elegir  $k$

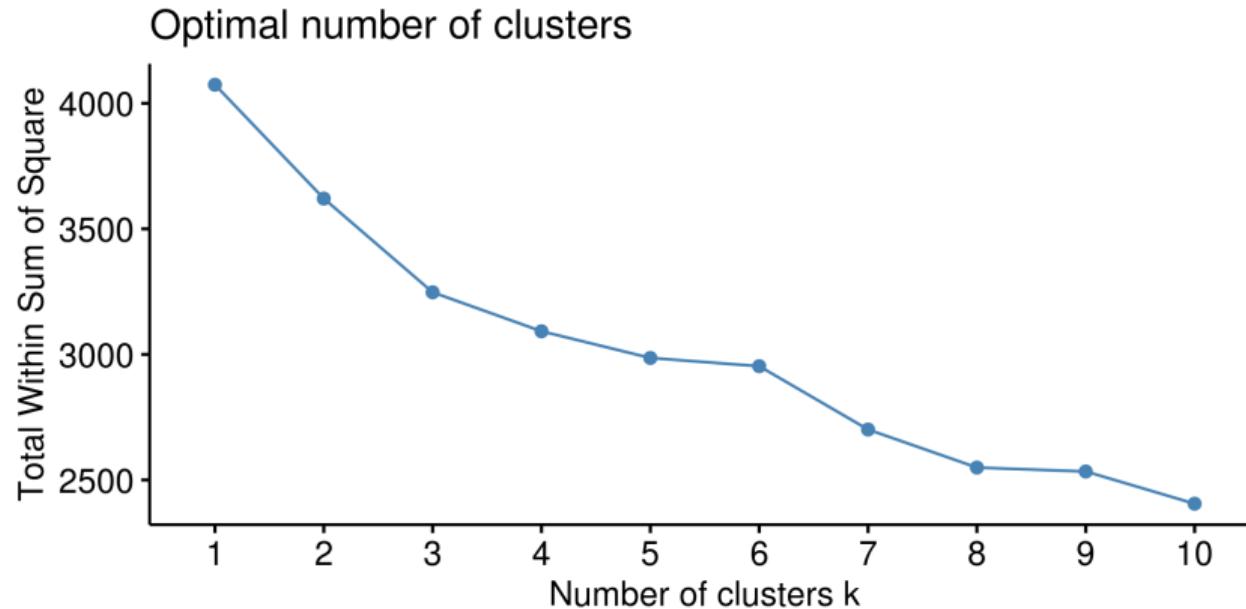
Método del codo:

```
fviz_nbclust(B, FUNcluster = pam, method = "wss", diss = d)
```

Revisemos los criterios para elegir  $k$

Método del codo:

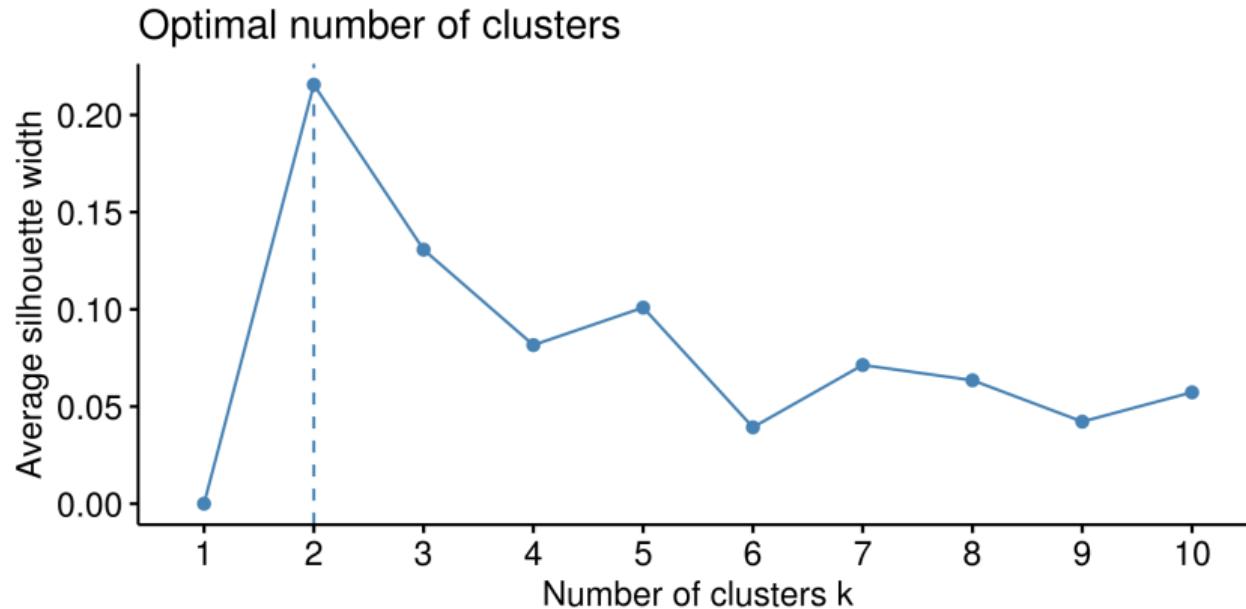
```
fviz_nbclust(B, FUNcluster = pam, method = "wss", diss = d)
```



Revisemos los criterios para elegir  $k$

Método de la silueta:

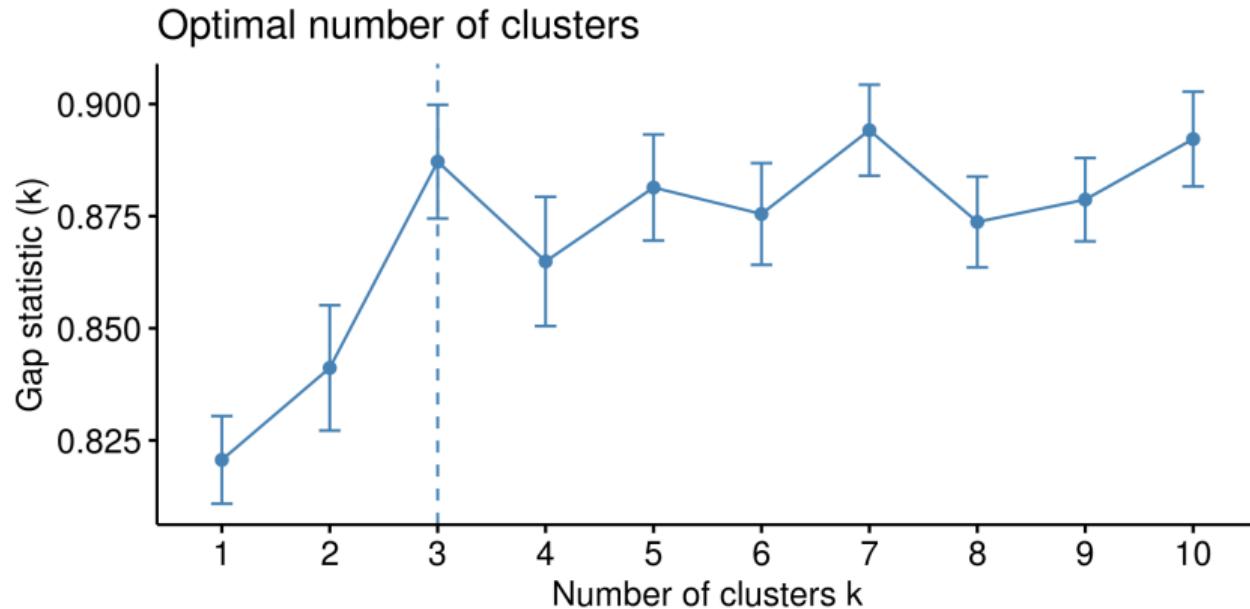
```
fviz_nbclust(B, FUNcluster = pam, method = "silhouette", diss = d)
```



Revisemos los criterios para elegir  $k$

Método del *gap statistic*:

```
fviz_nbclust(B, FUNcluster = pam, method = "gap", diss = d)
```



Y la votación de los otros treinta índices:

```
NbClust(data = B, diss = d, distance = NULL, method = "ward.D2")
```

```
*****
```

```
* Among all indices:
```

- \* 9 proposed 2 as the best number of clusters
- \* 8 proposed 3 as the best number of clusters
- \* 2 proposed 4 as the best number of clusters
- \* 2 proposed 7 as the best number of clusters
- \* 2 proposed 15 as the best number of clusters

```
***** Conclusion *****
```

```
* According to the majority rule, the best number of clusters is 2
```

```
*****
```

Y la votación de los otros treinta índices:

```
NbClust(data = B, diss = d, distance = NULL, method = "ward.D2")
```

```
*****
```

```
* Among all indices:
```

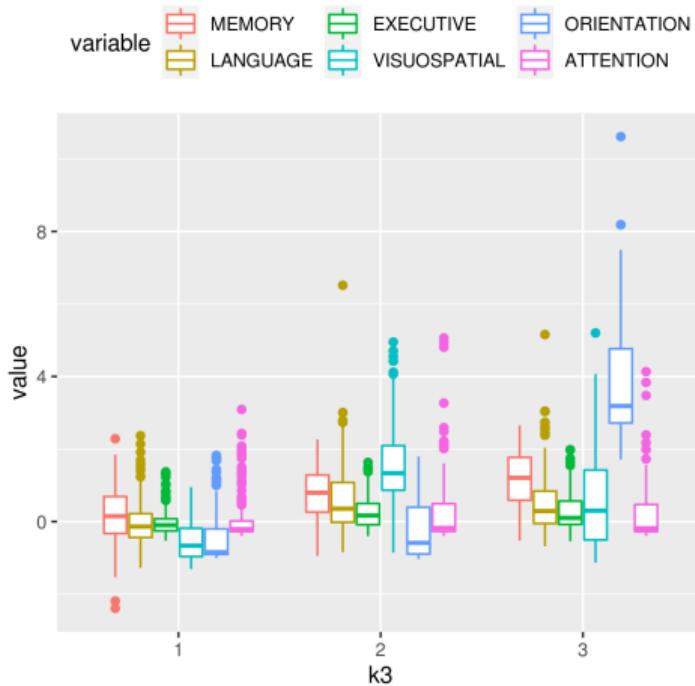
- \* 9 proposed 2 as the best number of clusters
- \* 8 proposed 3 as the best number of clusters
- \* 2 proposed 4 as the best number of clusters
- \* 2 proposed 7 as the best number of clusters
- \* 2 proposed 15 as the best number of clusters

```
***** Conclusion *****
```

```
* According to the majority rule, the best number of clusters is 2
```

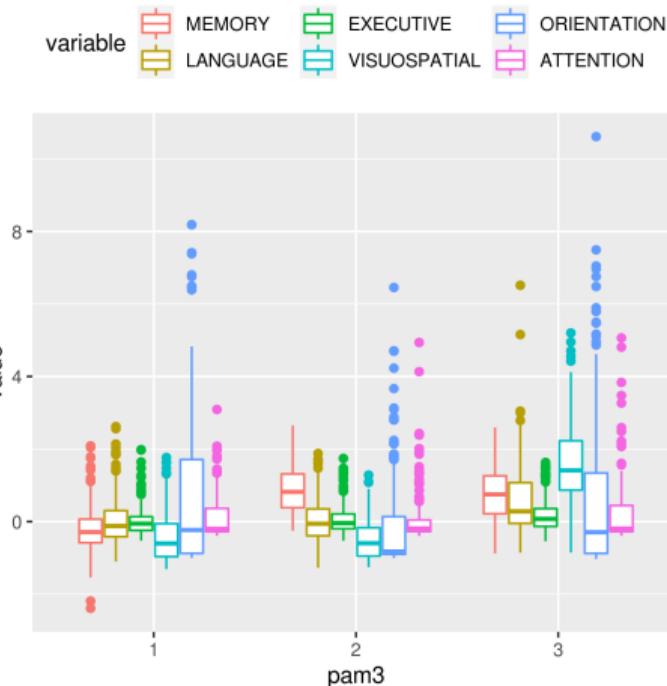
```
*****
```

# Comparación de resultados



1: 387, 2: 176, 3: 117

$k$ -means, distancia euclídea



1: 214, 2: 256, 3: 210

PAM, distancia de Mahalanobis

# let's **DISCUSS**



Gracias!

[dia.giraldo@gmail.com](mailto:dia.giraldo@gmail.com)