Semillas de trigo (**K-means**)

Preparación de los datos

Importando el CSV de los datos

Antes de comenzar con el análisis de los datos, debemos cargar el archivo de **comma-separated values** (**CSV**) de éstos en **R Studio**. Para poder acceder y descargar directamente el archivo en caso de que se desee efectuar un análisis más exhaustivo de los mismos en un futuro consultar https://data.world/databeats/seeds.

```
# Biblioteca para hacer la lectura del archivo.
library(readr)
# Leyendo el CSV dándole la ruta relativa.
Data = read.csv("../../data/seeds_dataset.csv")
# Visualizando el contenido del CSV.
View(Data)
```

```
## Warning in View(Data): X cannot set locale modifiers
```

Eliminando etiquetas de clase e identificadores

Dado que haremos un análisis comparativo por medio del algoritmo de aprendizaje de máquina no supervisado de **k-means**, vamos a eliminar para el conjunto sobre el cuál se estudiará la estructura de las características el valor de la variable objetivo (de salida), misma que no existe en el aprendizaje no supervisado al no haber como tal respuesta correctas e incorrectas.

```
# Tenemos una copia de los datos.
datos.caracteristicas = Data
# A la copia le quitamos las columnas que no son de interés.
datos.caracteristicas$ID <- NULL
datos.caracteristicas$seedType <- NULL
# Vemos cómo quedaron los nuevos datos.
View(datos.caracteristicas)
```

```
## Warning in View(datos.caracteristicas): X cannot set locale modifiers
```

Reescalamiento de las características

Con tal de evitar que algunas características de las semillas de trigo tengan más importancia que otras —por ejemplo, el área y el perímetro tienen valores considerablemente mayores a los de los coeficientes de asimetría—, procedemos a normalizar los valores.

```
# Normalizando los datos y luego visualizándolos.
datos.escalados <- as.data.frame(scale(datos.caracteristicas))
View(datos.escalados)
```

```
## Warning in View(datos.escalados): X cannot set locale modifiers
```

Creación de clusters

Fijando la semilla de aleatoriedad

Como estudiamos, el algoritmo de **k-means** tiene una componente aleatoria en cuando a la selección de los centroides iniciales sobre los cuales se comienza a iterar para ir agrupando los datos. Para que los resultados sean siempre consistentes y reproducibles.

```
# Fijando la semilla de aleatoriedad.
set.seed(80)
```

Especificación de los clusters

En este caso de entrada ya sabíamos que hay un total de tres diferentes tipos de semillas de trigo, así que podemos especificar que queremos un total de tres agrupamientos para después poder hacer el análisis comparativo de los resultados obtenidos con el algoritmo de **k-means** y las etiquetas de clase correctas que no se ocupan durante el mismo.

Sin embargo, posteriormente evaluaremos una forma de aproximar adecuadamente el número de clusters convenientes en el caso general.

```
# Ejecución del algoritmo k-means con 3 clusters y guardando el resultado en la var. resultados.
resultados <- kmeans(datos.escalados, centers = 3)
```

Obteniendo información de los clusters creados

Podemos acceder a la información que nos arrojan los clusters que fueron determinados por el algoritmo de agrupamiento **k-means** como son

- Cómo fueron las asignaciones de las observaciones (ejemplares) a cada cluster.
- La inercia total de los clusters formados.
- La inercia inter-cluster.
- La inercia intra-cluster (un valor para cada cluster formado).
- La suma total de las inercias intra-cluster.

A continuación vemos el resultado para cada uno de los aspectos antes descritos:

```
# Viendo las diferentes propiedades del resultado.
resultados$cluster
```

```
##   [1] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 2 3 2 2 2 2 2 3
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 2 2 1
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3
## [176] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 2 3 3 3 3 3 3 3 3
```

```
resultados$totss
```

```
## [1] 1463
```

```
resultados$betweenss
```

```
## [1] 1034.392
```

```
resultados$withinss
```

```
## [1] 139.5542 144.4586 144.5954
```

```r
resultados$tot.withinss
```

```
## [1] 428.6082
```

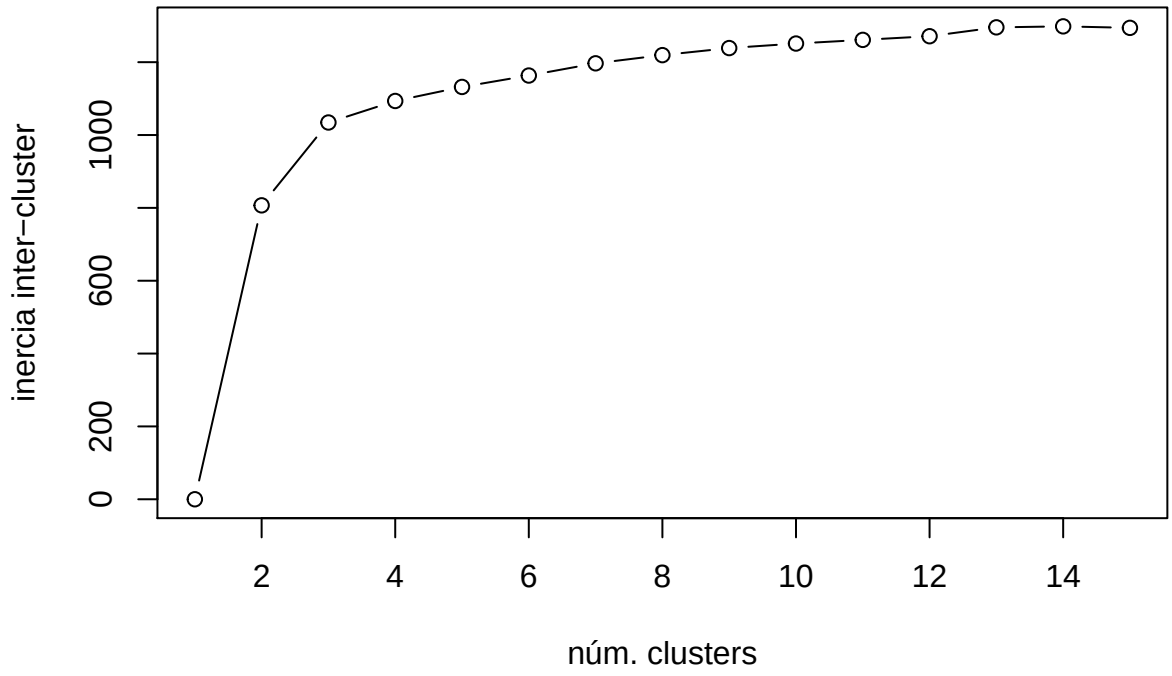
Obteniendo un aproximado al óptimo número de clusters

Como bien se mencionó en la clase, determinar de entrada cuántos clusters deberán formarse para que el agrupamiento sea óptimo es una tarea complicada pues como tal depende de lo que estemos buscando. Por esta razón conviene entonces graficar la inercia inter-clusters con un número variable de los mismos para ver en qué momento logra estabilizarse haciendo una exploración.

```r
# Almacenamos en un vector que contendrá las inercias inter-cluster cuando k = 1.
vector <- kmeans(datos.escalados, centers = 1)$betweenss

# Ahora hacemos lo mismo, con una secuencia de valores para k, con k desde 2 hasta 15.
# Nótese que el número en el que nos detenemos podría también no ser el adecuado.
for(i in 2:15) vector[i] <- kmeans(datos.escalados, centers = i)$betweenss

# Graficamos para tener una idea del k adeacuado.
plot(1:15, vector, type = "b", xlab = "núm. clusters", ylab = "inercia inter-cluster")
```



De esta gráfica podemos ver que el número de clusters logra estabilizarse aproximadamente alrededor de $k = 3$, así que sí tenía sentido la creación de los tres clusters como ya sabíamos para este caso en particular de antemano.

Resultados

Medias de observaciones por cluster

Para cada uno de los clusters que fueron conformados, podemos ver cuál es el valor promedio para cada una de las observaciones originales:

```r
# Crenado la tabla con los promedios para cada categoría de observación.
aggregate(datos.caracteristicas, by = list(resultados$cluster), mean)
```

```
##   Group.1     area perimeter compactness lengthOfKernel widthOfKernel
## 1       1 18.49537  16.20343   0.8842104       6.175687      3.697537
## 2       2 14.43789  14.33775   0.8815972       5.514577      3.259225
## 3       3 11.85694  13.24778   0.8482528       5.231750      2.849542
##   asymmetryCoefficient lengthOfKernelGroove
## 1             3.632373             6.041701
## 2             2.707341             5.120803
## 3             4.742389             5.101722
```

Comparativa con tipos de semilla reales

Ahora, como parte del análisis comparativo solicitado, vamos a contrastar los ejemplares que fueron agrupados en cada cluster con su clase (por medio de la etiqueta de clase de **seetType** con que contábamos en un principio).

Tabla comparativa

Primero consideremos una tabla que compare a las semillas originales con cómo fueron agrupados los ejemplares realmente por cada cluster.

```r
table(Data$seedType, resultados$cluster)
```

```
##
##      1  2  3
##   1  2 62  6
##   2 65  5  0
##   3  0  4 66
```

Podemos ver que de los setenta ejemplares de semilla que se tenían, la amplia mayoría fueron agrupados de acuerdo a su tipo de semilla de trigo real.

Lo que nos indica la tabla anterior es:

1. El primer tipo de semillas tuvo un total de $2 + 6 = 8$ agrupamientos incorrectos, pero un total de sesenta y dos semillas fueron todas argupadas en el segundo cluster.
2. Para el segundo tipo de semilla, no hubo ninguna que fuera colocada en el tercer cluster, pero hubo un total de cinco semillas incorrectamente colocadas en el segundo cluster y el resto en el primer cluster.
3. En el caso del tercer tipo de semilla de trigo, no hubo ninguna que fuera colocada en el primer cluster. Seseinta y seis de ellas se identificaron como del tercer cluster, mientras que las cuatro restantes fueron identificadas como pertenecientes al segundo cluster.

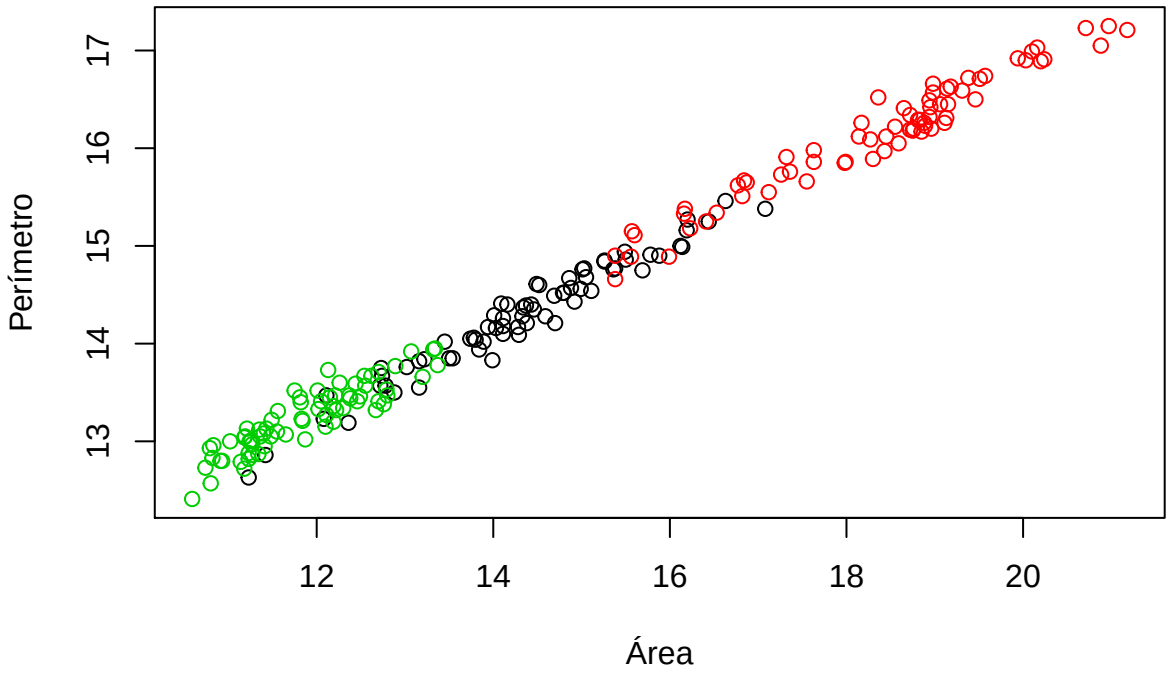
A partir de estos resultados, podemos concluir que el primer tipo de semilla de trigo es el más complicado de diferenciar, mientras que el tercer tipo es el más sencillo (siendo el que tuvo mayor éxito).

Más aún, el segundo tipo de semilla es fácil de contrastar con el tercer tipo.

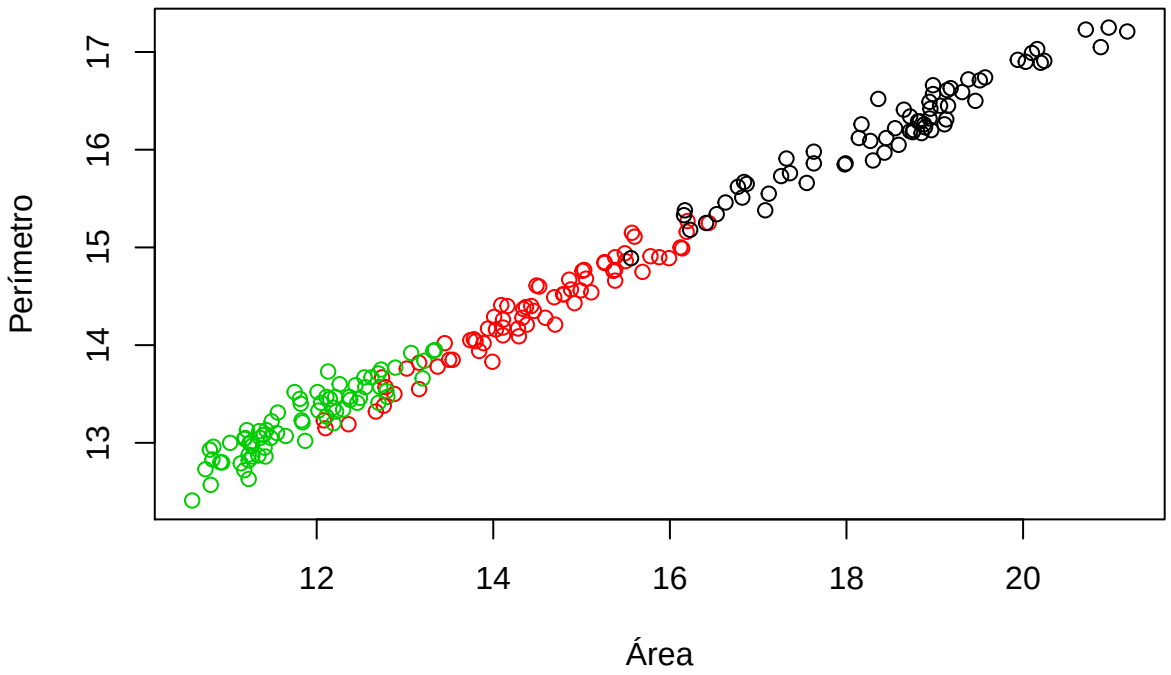
Graficación

Vamos a comparar el área y el perímetro de las semillas originales y ver una comparativa con la misma gráfica con los clusters obtenidos.

```r
# Gráfica de acuerdo a la etiqueta de clase correcta (la original).
plot(Data[c("area", "perimeter")], xlab = "Área", ylab = "Perímetro", col = Data$seedType)
```



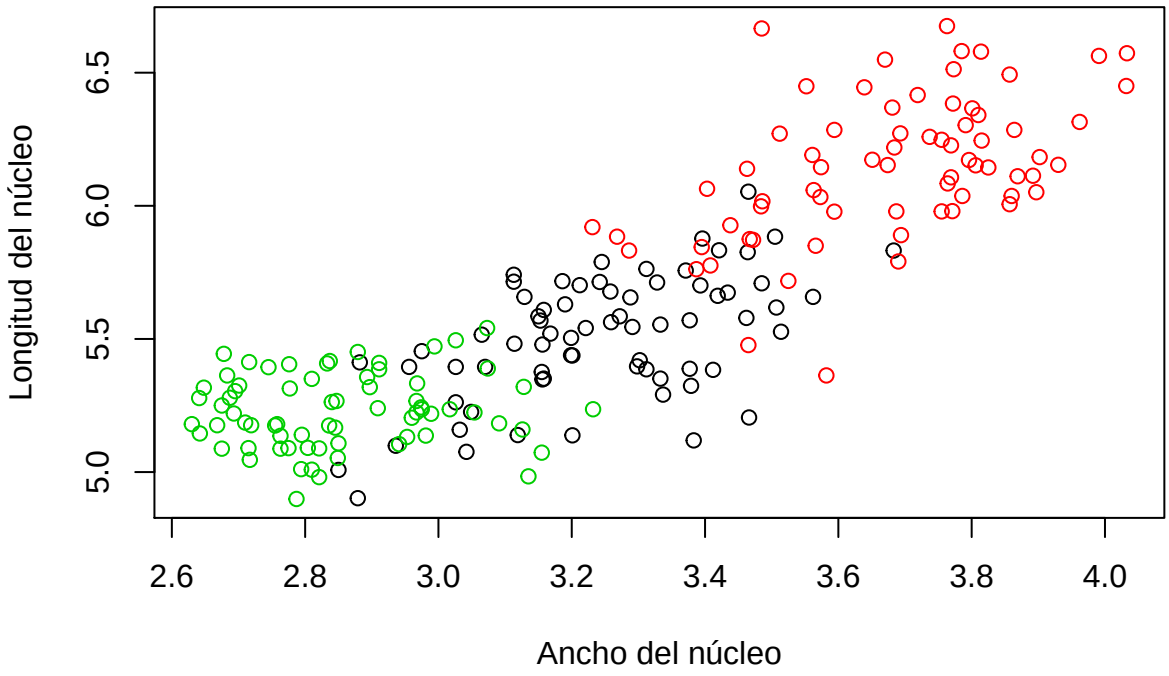

```r
# Gráfica de acuerdo al agrupamiento por clusters.
plot(Data[c("area", "perimeter")], xlab = "Área", ylab = "Perímetro", col = resultados$cluster)
```



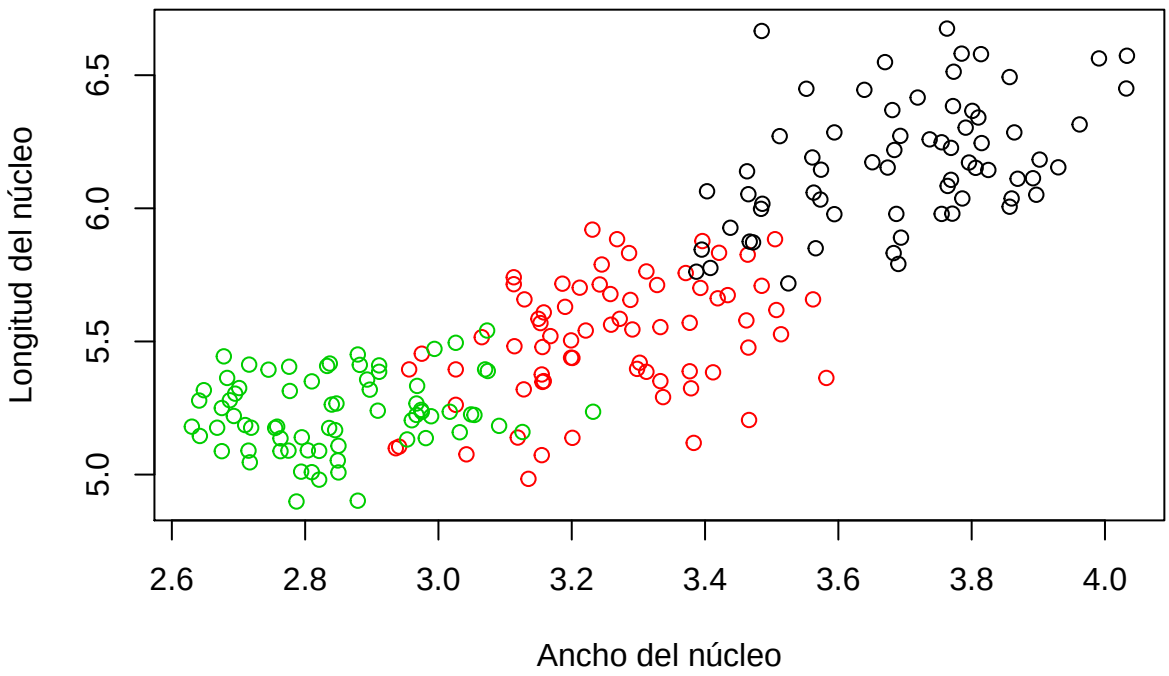
Podemos ver que los agrupamientos son muy parecidos (Los colores son distintos por motivos de cómo se toman en **R**).

Ahora hagamos la misma comparativa con respecto al ancho y longitud de los núcleos d elas semillas de trigo:

```r
# Gráfica de acuerdo a la etiqueta de clase correcta (la original).
plot(Data[c("widthOfKernel", "lengthOfKernel")], xlab = "Ancho del núcleo", ylab = "Longitud del núcleo
```




```r
# Gráfica de acuerdo al agrupamiento por clusters.
plot(Data[c("widthOfKernel", "lengthOfKernel")], xlab = "Ancho del núcleo", ylab = "Longitud del núcleo
```



De nueva cuenta podemos ver que los agupamientos por cluster se aproximaron bien a las clases originales.