CURSO: CC50 – ADMINISTRACION DE LA INFORMACION

CLASE: SESION #3 (PRACTICA)

TEMA: USO DE TABLAS, VECTORES Y LISTAS EN R / RSTUDIO

PROFESOR/A: Ing. PATRICIA REYES SILVA

En esta clase, trabajaremos con un conjunto de datos y lo conoceremos a detalle a partir de las instrucciones que R facilita para la exploración y visualización de los datos.

OBJETIVO PRINCIPAL

Realizar un análisis exploratorio básico en un conjunto de datos y crear visualizaciones utilizando las principales instrucciones que ofrece R.

COMPETENCIAS

- Familiarizarse con las instrucciones básicas en R para el uso de tablas (dataframes), vectores y listas.
- Aprender a leer y manipular conjuntos de datos
- Visualizar información relevante a partir del conjunto de datos analizado.

CASO DE ANALISIS

En esta práctica, haremos un análisis de datos exploratorio del famoso conjunto de datos **Iris**.

El conjunto de datos de **Iris** contiene cuatro características (largo y ancho de sépalos y pétalos) de 50 muestras de tres especies de Iris (Iris setosa, Iris virginica e Iris versicolor), en total, 150 observaciones. Estas medidas se utilizaron para crear un modelo discriminante lineal para clasificar las especies. Este conjunto de datos, creado por R.A. Fisher en el año 1936, se utiliza a menudo en ejemplos de minería de datos, clasificación y agrupación y para probar algoritmos.

Mayor información sobre este conjunto de datos se puede encontrar en el Repositorio de aprendizaje automático de la UCI – como Conjunto de datos Iris.

Como referencia, a continuación, se muestran imágenes de las tres especies de flores:

Iris Versicolor **Iris Setosa** **Iris Virginica**

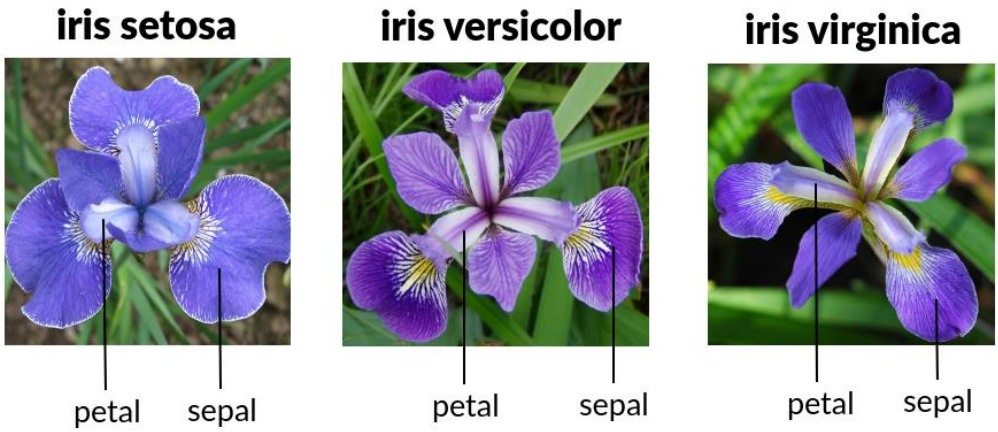
Los sépalos forman el cáliz de la flor y son normalmente de color verde y están en la base de la flor mientras que los pétalos son la parte más vistosa y aparente, de colores vivos que atraen a los insectos polinizadores.

CONJUNTO DE DATOS (DATA SET)

Es posible descargar los datos del dataset Iris ubicado en el Repositorio de aprendizaje automático de UCI - Conjunto de datos Iris, pero la biblioteca de conjuntos de datos en R ya lo contiene.

A continuación, se detalla la estructura del dataset Iris.

Variable	Descripción
Sepal.Length	Longitud del sépalo.
Sepal.Width	Ancho del sépalo.
Petal.Length	Longitud del pétalo.
Petal.Width	Ancho del pétalo.
Species	Especie a la que pertenece la flor: Iris **setosa**, Iris **virginica** e Iris **versicolor**



INSTRUCCIONES EN R / R STUDIO

Se realizarán las siguientes tareas desde la Consola en R:

I. CARGAR DATOS
II. INSPECCIONAR DATOS
III. VISUALIZACION GRAFICA

I. CARGAR DATOS

Escribimos en la Consola de R **iris**, que es lo mismo que ejecutar **print(iris)** y visualizaremos los datos del dataset.

```
> print(iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
```

II. INSPECCIONAR DATOS

Revisaremos los siguientes temas:

- Instrucciones básicas con data sets
- Selección de filas y columnas de un conjunto de datos
- Copia o eliminación de un conjunto de datos
- Creación / eliminación de columnas en un conjunto de datos
- Ordenamiento de filas dentro del conjunto de datos
- Creación de Vectores
  - Exploración de vectores
  - Selección de elementos de un vector
  - Creación de funciones
  - La función tapply
- Creación de Listas

1. Instrucciones básicas con data sets

Para explorar los datos contenidos en este o en cualquier otro data set, las instrucciones básicas son las siguientes:

Formato: instrucción (conjunto de datos)

Instrucción	Resultado
head(iris)	Visualizar primeras seis filas del dataset
tail(iris)	Visualizar primeras seis últimas filas del dataset
summary(iris)	Resumen estadístico de las columnas
str(iris)	Estructura textual del objeto
dim(iris)	Filas x columnas
nrow(iris)	Número de filas
ncol(iris)	Número de columnas
colnames(iris)	Se obtiene el nombre de las columnas

2. Selección de filas y columnas de un conjunto de datos

Si deseamos trabajar con un subconjunto de la tabla (filas y columnas determinadas), se utiliza el corchete []

Formato: conjunto de datos [filas o blanco, columnas o blanco]

Instrucción	Resultado
iris[1:7,]	Visualizar 7 primeras filas del dataset
iris[, 3:4]	Visualizar columnas 3 y 4 del dataset
iris[1:10, 3:4]	Visualizar 10 primeras filas de las columnas 3 y 4 del dataset

Dentro de los corchetes, también se admite el nombre de la columna en lugar del índice que dicha columna ocupa en el dataset. Lo operadores lógicos >, <, ==, &, etc. también son contemplados.

Instrucción	Resultado
iris[, "Species"]	Visualiza todos los valores de la columna llamada Species del dataset
iris$Species	Visualiza todos los valores de la columna llamada Species del dataset, equivale a iris[, "Species"]
iris[iris$Species == "setosa",]	Visualiza todas las filas que cumplan la condición que la especie sea “setosa” dentro del dataset
iris[iris$Species == "setosa" & iris$Petal.Length >= 1.5,]	Visualiza todas las filas que cumplan la condición que la especie sea “setosa” y que la longitud del petalo sea mayor o igual a 1.5

3. Copia o eliminación de un conjunto de datos

En muchas ocasiones no se desea trabajar con el dataset original, por ello, lo recomendable es crear una copia de este y trabajar sobre la copia.

En R para darle un nombre a un nuevo objeto, es recomendable no utilizar un numero al inicio del nombre, y de preferencia, utilizar "." o "_" como separador, cuando un nombre de objeto es compuesto por más de una palabra.

Copia:

```
> my.iris <- iris # donde my.iris es una copia de iris
    head(my.iris)

> head(my.iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

>
> ls() # lista de objetos que se mantienen en la memoria (solo los creados por nosotros)
```

Eliminación:

```
> rm(my.iris) # borra el objeto my.iris
> ls()
```

Los objetos nuevos creados aparecerán listados en el panel superior derecho de RStudio (si es que lo utilizas).

4. Creación / eliminación de columnas en un conjunto de datos

Crear columna:

```
> my.iris <- iris

> my.iris$Petal.Area <- my.iris$Petal.Length * my.iris$Petal.Width
```

Eliminar columna:

```
> my.iris$Petal.Area <- NULL
```

IMPORTANTE SABER:

- REFERENCIAR una columna que no existe la crea.
- CREAR una columna que ya existe, reemplaza a la anterior.
- ASIGNAR NULL a una columna existente la elimina.

5. Ordenamiento de filas dentro del conjunto de datos

R utiliza la función **order** para seleccionar ordenadamente una columna o más columnas (admite más de un argumento.

Por ejemplo, podemos crear un dataset llamado my.iris donde las filas estén ordenadas según la longitud de los pétalos.

```
> my.iris <- iris[order(iris$Petal.Length),]
```

6. Creación de Vectores

Crearemos dos vectores, uno numérico y otro categórico (del tipo factor, es decir, que considera niveles).

El vector numérico x, contempla número del 1 al 10

```
> x <- 1:10
```

El vector numérico **y**, está compuesto por las categorías (o niveles) de especies del dataset iris.

```
> y <- iris$Species
```

Observamos los vectores creados con la instrucción **ls()**

Se puede crear también secuencias de números enteros utilizando el operador ":" y para construir vectores arbitrarios, podemos usar la función de concatenación, "c" como se ejemplifica a continuación:

```
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> c (1:5, 5:1)
 [1] 1 2 3 4 5 5 4 3 2 1
[1] 1 2 3 4 5 5 4 3 2 1

> c(1, 5, -1, 4)
[1]  1  5 -1  4
> c("uno", "dos", "tres")
[1] "uno"  "dos"  "tres"
```

Todas estas operaciones que no son asignadas a variables simplemente sus resultados se visualizan por la consola.

Se utiliza la función **seq()** para reemplazar al operador ":"

```
> seq(1, 5)
[1] 1 2 3 4 5
```

Y la función **seq()** se relaciona estrechamente con la función **rep()**

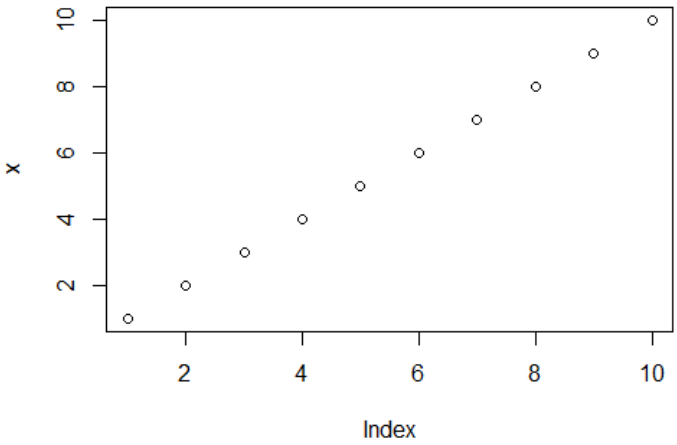
```
> rep(1:5, 5) #repite la secuencia n veces
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

> rep(1:5, each = 5) #repite cada element n veces
```

```
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
```

6.1. Exploración de vectores

Podemos aplicar las siguientes funciones, algunas de ellas ya utilizadas para inspeccionar data sets, y que también las utilizamos para ver el contenido de un vector: plot(vector), length(vector), table(vector), summary(vector), head(vector) y tail(vector)

```
> plot(x)
```




```
> length(x)
[1] 10

> table(y)
y
    setosa versicolor  virginica
        50         50         50
   50    50    50

> summary(y)
    setosa versicolor  virginica
        50         50         50

> head(x)
[1] 1 2 3 4 5 6

> tail(x)
[1]  5  6  7  8  9 10
```

Donde el vector “y” equivale a table(iris$Species)

```
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

6.2. Selección de elementos de un vector

- Utilizamos el corchete [] para seleccionar elementos de un vector. A diferencia de lo que ocurre con los data sets (tablas), como los vectores son objetos unidimensionales, sin coma. En este caso, el corchete además de los índices de los elementos que se quieren extraer, admite además utilizar condiciones lógicas, etc.

```
> x <- x^2

> x
 [1]   1   4   9  16  25  36  49  64  81 100

> x[1:3]
[1] 1 4 9

> x[c(1,3)] #selecciona los índices indicados
[1] 1 9

> x[x > 25]
[1]  36  49  64  81 100

> x[3:1] #seleccion de elementos de forma inversa
[1] 9 4 1

> x[-(1:2)] #remueve los dos primeros elementos
[1]   9  16  25  36  49  64  81 100

> x[-length(x)] #remueve el elemento equivalente a la posición señalada
[1]  1  4  9 16 25 36 49 64 81
```

- Utilizando el corchete también podemos seleccionar elementos de un vector por nombre. En R se pueden asociar nombres a los elementos de un vector. Lo hace automáticamente, por ejemplo, table (para poder saber a qué etiqueta corresponde cada conteo):

```
> w <- table(iris$Species)

> w["setosa"]
setosa
    50

> w[c("setosa", "virginica")]

   setosa virginica
       50        50
```

- Como el nombre de las columnas de una tabla también es un vector, para cambiar el nombre de una columna podemos hacer lo siguiente:

```
> my.iris <- iris # una copia de iris

> colnames(my.iris)[5] <- "Especie"


> colnames(my.iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Especie"
```

- Igual sucede con los nombres de un vector, pero en este caso, la función correspondiente a utilizar es **names**:

```
> z <- table(iris$Species)

> names(z)
[1] "setosa"     "versicolor" "virginica"

> names(z)[1] <- "Tipo 1"

> names(z)
[1] "Tipo 1"     "versicolor" "virginica"
```

- Con el corchete, además de seleccionar, podemos cambiar el contenido de los elementos seleccionados de un vector. Por ejemplo:

```
> z <- 1:10

> z
 [1]  1  2  3  4  5  6  7  8  9 10

> z[z < 5] <- 100

> z
 [1] 100 100 100 100   5   6   7   8   9  10
```

- Por último, cuando se quiere obtener una serie de elementos al azar dentro de un vector, utilizamos la función **sample**:

```
> x <- 1:10

> x
 [1]  1  2  3  4  5  6  7  8  9 10

> sample(x,4)
[1] 9 3 1 7

> sample(x,50)
Error in sample.int(length(x), size, replace, prob) :
  cannot take a sample larger than the population when 'replace = FALSE'

> sample(x, 50, replace = TRUE)
 [1]  5  1  5  4  4  2  4  4  5  5  8  8 10  2  9 10  8  2  6 10  5  3  1
2  7  2  2  2  6 10  8  5
[33] 10  3  8 10  2  2  1  2 10  8  2  3  7  5  8 10  8  6
```

6.3. Creación de funciones

En R, hay muchas funciones que pueden aplicarse a un vector, pero las más comunes que pueden aplicarse a vectores numéricos son:

```
> x <- 1:10

> mean(x)
[1] 5.5

> max(x)
[1] 10

> median(x)
[1] 5.5

> sum(x)
[1] 55

> prod(x)
[1] 3628800
```

- **Función suma de cuadrados**

No existe en R una función que sume los cuadrados de unos números. Tendríamos que crearla así:

```
> x <- 1:10

> suma_cuadrados <- function(x) sum(x*x)

> suma_cuadrados(x)
```

```
[1] 385
```

➢ **Función media**

Cuando el cuerpo de la función sea de carácter más complejo, hay que encerrarlo en llaves {}

```
media <- function(x){
    longitud <- length(x)
    suma <- sum(x)
    suma / longitud
}
```

6.4. La función tapply

La función **tapply** es una operación similar a la conocida en SQL como group by.

```
> tapply(iris$Petal.Length, iris$Species, mean)
    setosa versicolor  virginica
     1.462      4.260      5.552
```

En el ejemplo, **tapply** aplica la función mean a un vector, la longitud del pétalo, pero de especie en especie. Esta función respondería a la pregunta: ¿cuál es la media de la longitud del pétalo por especie?

7. Creación de Listas

La mayoría de los datos con los que se trabaja son estructurados: los dataframes o tablas contienen datos estructurados, esto significa que las columnas son del mismo tipo, todas tienen la misma longitud, etc. Sin embargo, cada vez se hace más necesario trabajar con datos desestructurados en la ciencia de datos. Por ello, el uso de las listas, nos ayudan a contener y manipular los datos desestructurados para facilitar el análisis.

Las listas son objetos de R. Por ejemplo, el data set iris, como todas las tablas en R, son **listas**.

```
> is.list(iris)
[1] TRUE
```

➢ **Creación de listas**

Una lista se crea con la función list()

```
> x <- list("a" = 2.5, "b" = TRUE, "c" = 1:3)
> x
$a
[1] 2.5

$b
[1] TRUE
```

```
$c
[1] 1 2 3
```

La lista es una estructura de datos que tiene componentes de tipos de datos mixtos.

Un vector que tiene todos los elementos del mismo tipo se llama vector atómico, pero un vector que tiene elementos de diferente tipo se llama lista.

Podemos verificar si es una lista con la función typeof () y encontrar su longitud usando length (). A continuación, se muestra un ejemplo de una lista que tiene tres componentes, cada uno de los cuales tiene un tipo de datos diferente.

```
> typeof(x)
[1] "list"
> length(x)
[1] 3
```

Su estructura la examinamos con str()

```
> str(x)
List of 3
 $ a: num 2.5
 $ b: logi TRUE
 $ c: int [1:3] 1 2 3
```

En este ejemplo, a, b y c se denominan etiquetas, lo que facilita la referencia a los componentes de la lista.

Sin embargo, las etiquetas son opcionales. Podemos crear la misma lista sin las etiquetas de la siguiente manera. En tal escenario, los índices numéricos se utilizan de forma predeterminada.

```
> x <- list(2.5,TRUE,1:3)
> x
[[1]]
[1] 2.5

[[2]]
[1] TRUE

[[3]]
[1] 1 2 3
```

- **Accediendo a los componentes de una lista**

Se puede acceder a las listas de manera similar a los vectores. Se pueden utilizar vectores enteros, lógicos o de caracteres para la indexación. Consideremos una lista como sigue.

```
> x <- list(name = "John", age = 19, speaks = c("English", "French"))
> x
$name
```

```
[1] "John"

$age
[1] 19

$speaks
[1] "English" "French"
```

a) Índice usando un vector entero

```
> x[c(1:2)]
$name
[1] "John"

$age
[1] 19
```

b) Usando un entero negativo para excluir el segundo componente

```
> x[-2]
$name
[1] "John"

$speaks
[1] "English" "French"
```

c) Índice usando vector lógico

```
> x[c(T,F,F)]
$name
[1] "John
```

d) Índice usando vector de caracteres

```
> x[c("age","speaks")]
$age
[1] 19

$speaks
[1] "English" "French"
```

- **Modificando los componentes de una lista**

Podemos cambiar los componentes de una lista mediante la reasignación. Podemos elegir cualquiera de las técnicas de acceso a componentes discutidas anteriormente para modificarlo.

```
> x[["name"]] <- "Patricia"; x
$name
[1] "Patricia"

$age
[1] 19
```

```
$speaks
[1] "English" "French"
```

- **Adicionando componentes a una lista**

Agregar nuevos componentes es fácil. ¡Simplemente asignamos valores usando nuevas etiquetas y listo!

```
> x[["married"]] <- TRUE
> x
$name
[1] "Pat"

$age
[1] 19

$speaks
[1] "English" "French"

$married
[1] TRUE
```

- **Eliminando componentes a una lista**

Podemos eliminar un componente asignándole NULL.

```
> x[["age"]] <- NULL
> str(x)
List of 3
 $ name   : chr "Pat"
 $ speaks : chr [1:2] "English" "French"
 $ married: logi TRUE
> x$married <- NULL
> str(x)
List of 2
 $ name  : chr "Pat"
 $ speaks: chr [1:2] "English" "French"
```

INFORMACION IMPORTANTE

Las tablas en R son básicamente listas de vectores de la misma longitud. Las listas también son contenedores genéricos de datos, donde se encapsulan información heterogénea.

Pero una lista también puede contener otras listas que, a su vez, pudieran contener otras, etc. Es decir, las listas permiten almacenar árboles de información. Los árboles de información son cada vez más importantes en las aplicaciones. De hecho, los ficheros .xml o .json disponen su información en árboles, como muestra el siguiente ejemplo de un fichero .json y el mismo en formato .xml:

Texto en JSON

```
{"menu": {
  "id": "file",
  "value": "File",
```

```
 "popup": {
   "menuitem": [
     {"value": "New", "onclick": "CreateNewDoc()"},
     {"value": "Open", "onclick": "OpenDoc()"},
     {"value": "Close", "onclick": "CloseDoc()"}
   ]
 }
}}
```

El mismo texto en XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

El fichero anterior tiene un elemento llamado **menu**, que contiene a su vez tres elementos: **id**, **value** y **popup**, y donde **popoup** contiene a su vez listas de un elemento llamado **menuitem (**contenedor de vectores con identificadores de valor `New`, `Open` y `Close`).

En la siguiente práctica correspondiente a la Sesión #4, trabajaremos con archivos .JSON y XML extraídos desde distintas fuentes de datos.

III. VISUALIZACION GRAFICA

1. Gráficos básicos en R

Los gráficos nos permiten una inspección visual y rápida de conjuntos de datos, tarea fundamental durante todo proceso de análisis de datos.

A continuación, representaremos de forma gráfica:

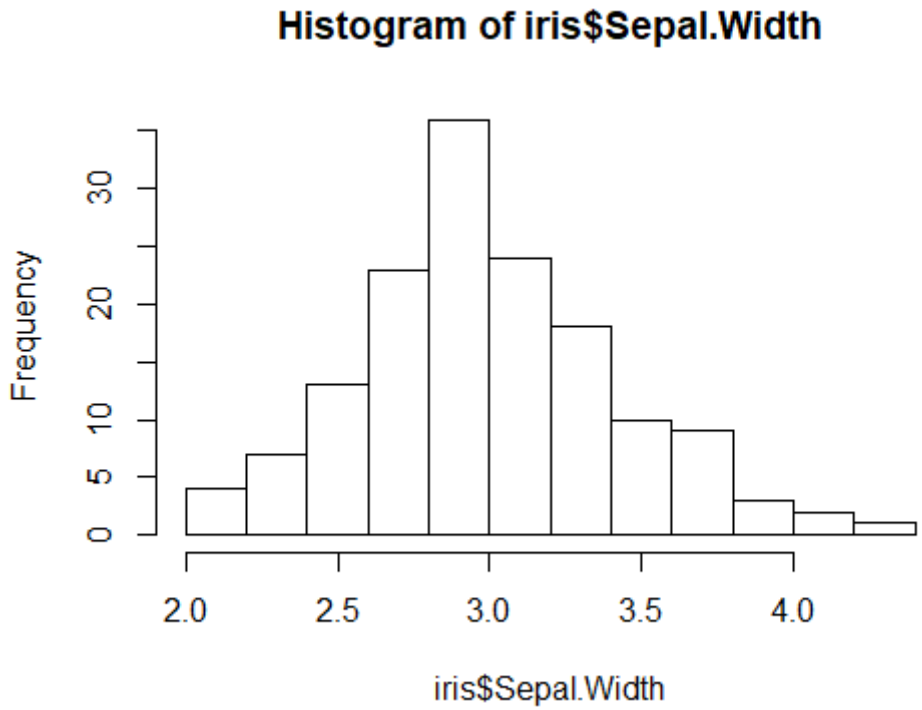
- Una variable continua (histogramas)

- Una variable categórica (grafico de barras)
- La relación entre dos variables continuas (gráfico de dispersión)
- La relación entre una variable continua y otra categórica (diagrama de cajas - boxplot).

1.1. Gráfica de variables continuas

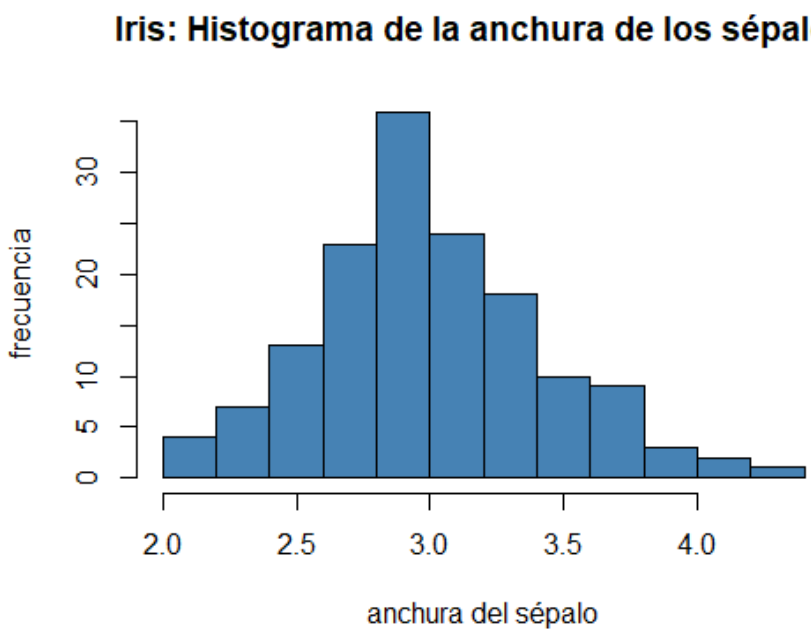
Para la representación gráfica de las variables continuas, utilizamos los **histogramas**. En R, para representar el histograma de la columna Sepal.Width de iris se construye con la instrucción **hist**.

```
> hist(iris$Sepal.Width)
```



Pero los gráficos pueden ser modificados para incluir títulos, etiquetas, colores, etc. Por ejemplo,

```
hist(iris$Sepal.Width, main = "Iris: Histograma de la anchura de los sépalos",
      xlab = "anchura del sépalo", ylab = "frecuencia",
      col = "steelblue")
```



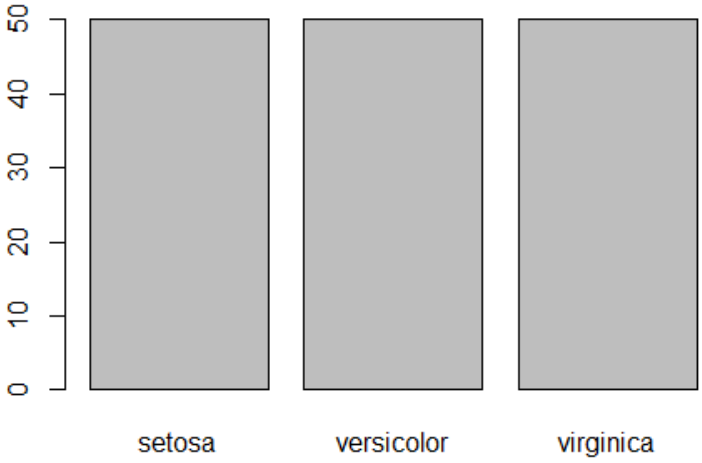
1.2. Gráfica de variables categóricas

En los conjuntos de datos, coexisten variables continuas y categóricas. Para visualizar, por ejemplo, la distribución (o frecuencia) de cada una de las categorías, se utilizan los **diagramas de barras**, con la función **barplot** de R.

La función **barplot** no muestra directamente las frecuencias de una variable categórica, se vale de la función table para calcularlas previamente.

En la siguiente grafica de barras se muestra cómo en el dataset Iris existe el mismo número de observaciones de cada especie:

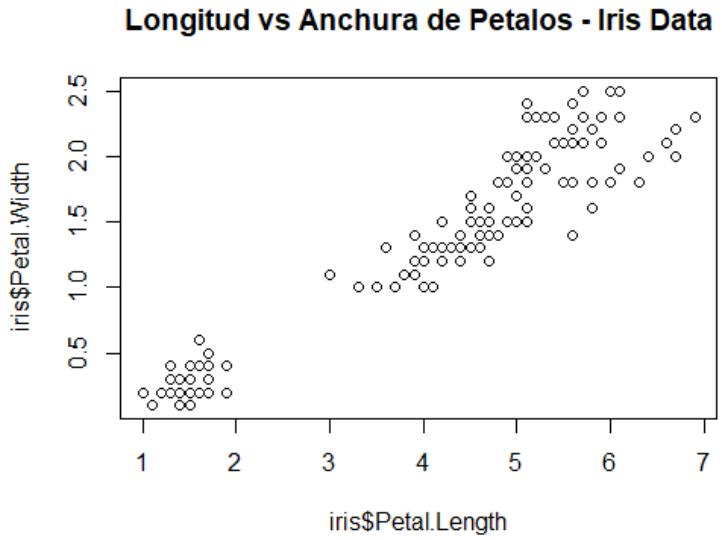
```
> barplot(table(iris$Species))
```



1.3. Gráfica de la relación entre dos variables continuas

Los **gráficos de dispersión** muestran la relación entre dos variables numéricas. Esta representación gráfica es muy común, porque los aspectos más interesantes de los datos se revelan no examinando las variables independientemente sino en relación con otras.

```
> plot(iris$Petal.Length, iris$Petal.Width, main="Edgar Anderson's Iris Da
ta")
```



El gráfico muestra la existencia de observaciones donde la longitud de los pétalos de un grupo de iris es de 1 a 2 cm y su anchura menor a 1 cm. Por otro lado, es mayor el número de observaciones de iris cuya longitud de pétalo es de 3 a 7 cm y la anchura es mayor a 1 cm.

1.4. Gráfica de la relación entre una variable continua y otra categórica

La distribución de una variable continua en función de una variable categórica es representada gráficamente en R a través de los **diagramas de cajas** (boxplot). Al igual que los histogramas, resumen la distribución de una variable continua, pero los diagramas de caja utilizan una representación más esquemática: una caja y unos segmentos que acotan las regiones donde la variable continua concentra el grueso de las observaciones.

Utilizando diagramas de caja, podemos por ejemplo visualizar la distribución de la anchura del sépalo en iris en función de la especie:

```
> boxplot(iris$Sepal.Width ~ iris$Species, col = "gray", main = "Especies
de iris\nsegún la anchura del sépalo")
```