```
library(dplyr)
library(magrittr)
```

```
Curso_R %>% filter(city == 'Corrientes')
```

BIENVENIDOS AL TALLER

APRENDER R DESDE CERO

Lunes 12 de agosto de 2019.

HOY HABLAMOS SOBRE...

PARTE 1: Buenas prácticas

Tipo de datos (objetos)

Datos Atómicos

Vectores

Matrices

Factores

Dataframes

Elementos de la sintaxis de R/ Coerción

PARTE 2: Estructuras de Control y Funciones

If-else

For

While

Funciones.

La importancia de trabajar con proyectos

Importación de Datos en R

Exploración básica de datos

PARTE 1

BUENAS PRÁCTICAS

1. **Llamar todos los paquetes al inicio de nuestro script**. De este modo, cuando más adelante volvamos a trabajar con el script (u otra persona quiera ejecutarlo) queda claro desde el principio cuáles paquetes se utilizan (y si sería necesario descargarlos, en caso de no tenerlos).
2. **Comentar nuestro script.** En R podemos agregar comentarios anteponiendo un **#** al fragmento que cumple esa función. En una línea de nuestro script, todo lo que está después de un **#** no se ejecuta como código.

¿POR QUÉ ES IMPORTANTE COMENTAR NUESTRO SCRIPT?

- Porque así podemos recordar qué es lo que un determinado fragmento de código hace, o podemos dejar registro de por qué realizamos algo de una determinada manera. Esto es muy útil para que nuestro futuro yo (o las personas con las que compartiremos nuestro script) entiendan lo que hicimos.

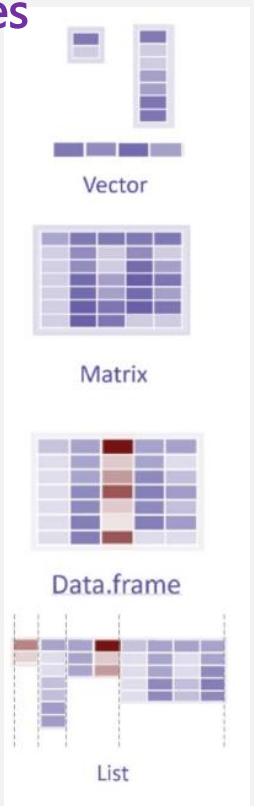
- Los comentarios también son muy útiles cuando estamos aprendiendo! Probablemente en el futuro ya no necesites indicar que “library()” sirve para cargar paquetes, pero por ahora es una buena manera de ir registrando qué es lo que hace cada función.

- También es útil agregar al inicio de nuestro script una descripción de su objetivo o del contexto en que lo escribimos.

TIPO DE OBJETOS

Unidimensionales

- **Vectores:** secuencia de elementos de datos del mismo tipo.
- **Factores:** ***vector*** utilizado para variables categóricas con número definido de niveles.
- **Matrices:** ***vectores columna*** de 2 dimensiones en la que cada elemento es del mismo tipo.
- **Arreglo (array):** similar a una matriz pero puede tener más de 2 dimensiones.
- **Dataframes:** ***lista*** en la que cada columna puede ser de diferente clase pero de igual longitud.
- **Listas:** conjunto indexado de objetos donde los objetos pueden ser de ***diferente clase y longitud***. Puede pensarse como un ***contenedor de objetos*** que pueden ser: vectores, factores, matrices, arreglos, dataframes y listas.



OPERADORES

- Las operaciones pueden ser agrupadas usando paréntesis, y asignadas a variables de manera directa.

Aritméticos	Comparativos o relacionales	Lógicos
+ adición **-** sustracción ***** multiplicación **/** división **~** potencia **%%** modulo **%/%** división de enteros	**<** menor que **>** mayor que **<=** menor o igual que **>=** mayor o igual que **==** igual **!** Diferente	**!** X 'NO' lógico **x & y** 'Y' lógico **x && y** idem **x | y** 'O' lógico **Xor (x,y)** O exclusivo

EN R TODOS LOS DATOS SON OBJETOS

DATOS ATÓMICOS

Pueden ser de diferentes tipos: reales, enteros, complejos o booleanos. La función ***class ()*** *nos dice el tipo de dato.*

#Numéricos (reales)
monto <- 2.5
class (monto)

#Enteros
cantidad <- 1L
class (cantidad)

#Complejos
z <- 2+2i
class (y)

#Caracteres
nombre<- “Patricia"
class (nombre)

#Lógicos o booleanos
es_estudiante <- TRUE
class (es_estudiante)

Al agregar la letra L final, R entiende que se trata de un número entero.

VECTORES

Son conjuntos de datos que pueden ser numéricos, lógicos o caracteres.

Se crea con la función **c ( )**

#Un vector de tipo numérico

```
edad <- c (50, 30, 20)
edad
```

#Un vector de tipo lógico o booleano

```
es_casado <- c (T,F,F,T,T)
print (es_casado)
```

#Un vector de cadena de caracteres

```
nombres <- c ("Julia", "Juan", "María")
print (nombres)
```

Con la función c( ) también se puede concatenar varios vectores

```
datos <- c (edad, nombres, es_casado)
datos
```

VECTORES

Podemos crear *secuencias* de distintas formas

#Mediante el operador :

```
seq1 <- 1:20
seq1

seq2 <- 40:20
seq2
```

Permite crear una secuencia creciente o decreciente

#Mediante la función seq( )

```
seq3 <- seq (1, 9, by=2)
seq3
```

Indexación
#Mediante el operador [ ]

```
v <- c (8, 5,2,1)
v[2]
```


Me permite acceder al segundo elemento

Con length( ) se puede saber la longitud de un vector

```
Length (seq1)
Length (seq3)
```

MATRICES

Desde el punto de vista del lenguaje, una matriz es un **vector** *con atributo adicional:* ***dim***. Es decir, es un vector de 2 dimensiones, a saber: el número de renglones o filas y el número de columnas.

Las matrices por defecto se crean de columna a columna, empezando por fila superior izquierda, completando hasta abajo hasta alcanzar la dimensión de filas, y luego sigue agregando las columnas siguientes de a una por vez hacia la derecha. En caso, debo especificar 'byrow'

```
mat3 <- matrix(11:30, nrow=5, ncol=4, byrow=True)
```

MATRICES

#Por medio de la función dim

```
dim (matriz) <- c (5:4)
matriz <- c (11:30)
matriz
class (matriz)
```

#Por medio de la función matrix()

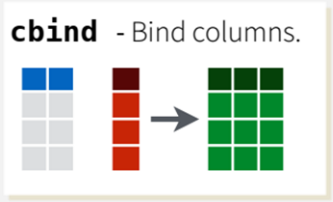
```
matriz2 <- matrix(11:30, nrow=5, ncol=4)
```

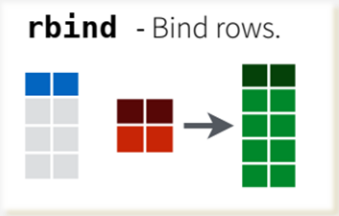
#Por medio de cbind ( ) y rbind ( )

```
a <- c (6,9,12)
b <- c (7,10,13)

c <- cbind (a,b)  añade una columna

d <- rbind (a,b)   añade una fila
```





MATRICES: INDEXACIÓN

Para acceder a los elementos de una matriz

#Mediante el operador [ ] ← Se puede acceder mediante elemento de la matriz como un vector

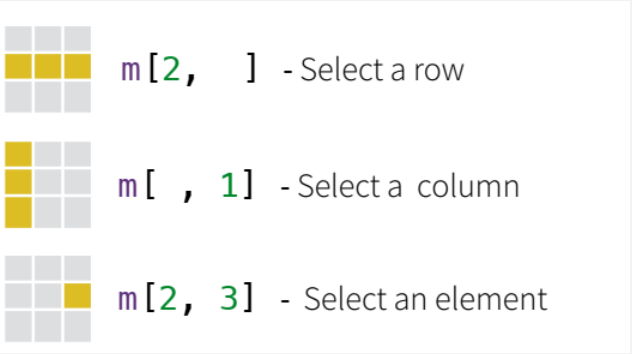
```
mat2 <- matrix(11:30, nrow=5, ncol=4)

m <- matriz2[3,2]
k <- matriz8]
```

 # Para Listas, mediante el operador [[ ]]

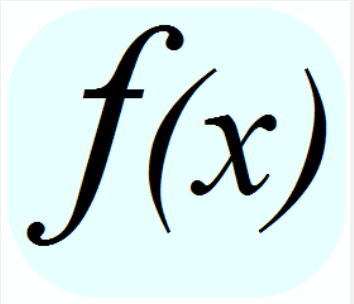
```
n <- matriz2 [[2]]

Y <- matriz2 [[2,3]]
```



ALGUNAS FUNCIONES DE MATRICES

dim	devuelve las dimensiones de una matriz
dimnames	devuelve el nombre de las dimensiones de una matriz
colnames	devuelve el nombre de las columnas de una matriz
rownames	devuelve el nombre de las filas de una matriz
mode	devuelve el tipo de datos de los elementos de una matriz
length	devuelve el número total de elementos de una matriz
is.matrix	devuelve T si el objeto es una matriz, F si no lo es
[,]	accede a elementos dentro de la matriz
apply	Aplica una función sobre las filas o columnas de una matriz
cbind	Añade una columna a una matriz dada
rbind	Añade una fila a una matriz dada



FACTORES

Son cadena de caracteres que se utilizan para nombrar cosas u objetos.

#Por medio de la función c ( )

```
persona <- c ("Pedro", "Juan", "Silvia", "María")
mes.nac <- c ("Enero", "Febrero", "Marzo", "Abril")

print (persona[2]); print (mes.nac[2])
```

#Es posible concatenar factores mediante la función paste( )

```
paste (persona[2],"nació en el mes de", mes.nac[2])
```

#Indexación mediante [ ]

Al ser una estructura unidimensional como los vectores, es factible usar el operador[ ] para indexar.

```
persona [1]
mes.nac [4]
```

LISTAS

Una lista es una clase de dato que puede contener cero o más elementos, **cada uno de los cuales puede ser de una clase distinta.**

Esto no ocurre con las matrices y los arrays que tienen elementos de igual clase.

#Por medio de la función list()

```
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
```

#Indexación mediante [ ], [[ ]] y $

```
list_data[1]
```

← *Selecciona una sublista*

```
list_data[[3]]
```

← *Selecciona un único elemento*

#Para utilizar $, los elementos deben tener un nombre.

```
list_2<-list(ciudades=c("Corrientes", "Resistencia"), edades=c(20,30))

list_2$edades
```

DATAFRAMES

Un dataframe es una **lista** cuyos componentes pueden ser vectores, matrices o factores, con la única salvedad de que las filas y columnas coincidan en todos sus componentes.

Tiene apariencia de una tabla.

#Por medio de la función data.frame()

```
x <- data.frame (sitios=1:4, muestreado = c(T,F,F,T))
```

#En R hay pre-cargados algunos dataframes

Ejemplo: mtcars e iris.

#En general, importamos los dataframes mediante read.table( ) y read.csv().

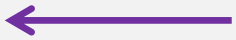
DATAFRAMES: INDEXACIÓN

#Mediante el operador [ ]

mtcars[1] mtcars [2,] mtcars[,2]

También permite acceder a un elemento concreto
mtcars[3,2]

#Mediante el operador $



Sólo para listas y dataframes

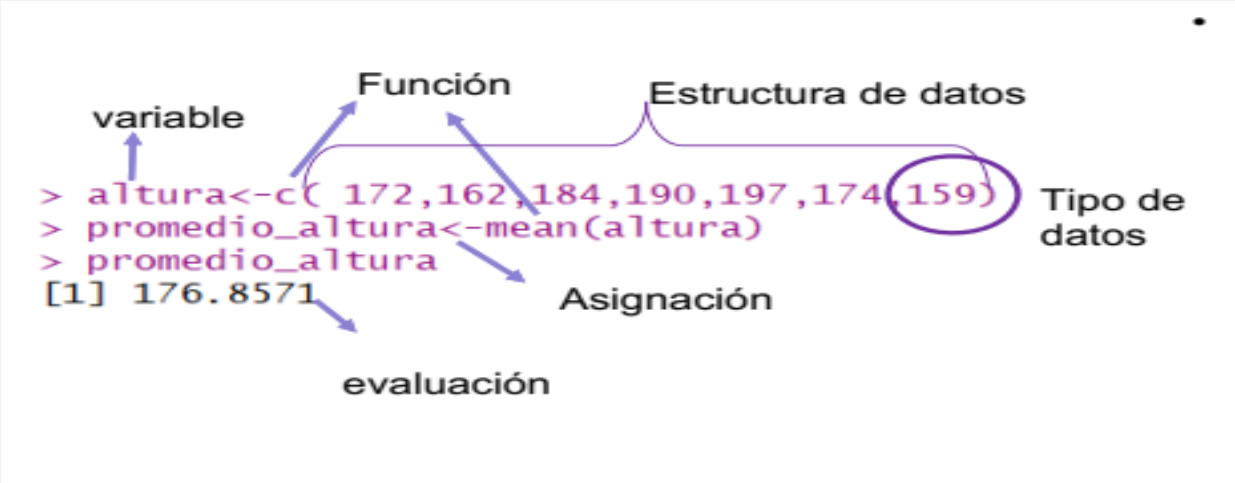
mtcars$hp
mtcars$w

#Podemos modificar una columna

mi.mtcars <- mtcars
mi.mtcars$hp <- mtcars$hp * 100

ELEMENTOS DE LA SINTAXIS DE R

- **Asignación**
- **Tipo de datos**
- **Estructura de datos**
- **Operadores**
- **Funciones**
- **Variables globales vs locales o temporales**



ELEMENTOS DE LA SINTAXIS DE R

Coerción:

- **Implícita**: R interpreta automáticamente que tipo de datos son cuando lee datos de tipo character, numeric o logical. Si lee un dato y no lo puede asignar, pone NA y avisa con un *warning.*

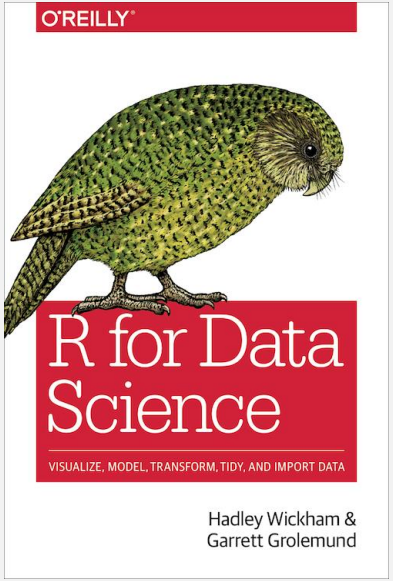
✓ x <- 1.5
✓ is.numeric(x)

- **Explícita**: se le indica a R que interprete un dato de determinada manera.

✓ x <-1.03
✓ class(x)
✓ y <- as.integer(x)
✓ class(y)

PARTE 2

Libro R para Ciencia de Datos



http://r4ds.had.co.nz/
https://github.com/hadley/r4ds

ESTRUCTURAS DE CONTROL

Las estructuras de control nos permiten controlar el flujo de ejecución de una secuencia de comandos. De este modo, podemos poner “lógica” en el código de R y lograr así reutilizar fragmentos de código una y otra vez.

Permiten evaluar las entradas, y ejecutar código diferente según ciertos casos.

- #if /else

Permite decidir si ejecutar o no un fragmento de código en función de una condición.

- #for

Ejecuta un bucle una cantidad fija de veces.

- #while

Ejecuta un bucle mientras sea verdadera una condición.

FUNCIONES

- Una función es un conjunto de instrucciones para realizar una tarea especifica
- Puede aceptar argumentos o parámetros
- Puede devolver uno o más valores o ninguno
- Usamos tanto las funciones que trae R preinstaladas como las que agregamos al cargar paquetes, o creando funciones propias

FUNCIONES

Constan de:

- lista de argumentos (arglist)
- código (body)
- entorno en el cual son válidas las variables que se crean y usan para realizar la acción de la función

```
Mifuncion <- function( arg1,arg2,...)
{ instrucciones
return (objeto)
}
```

Ejemplo:

sumacuadrado <- function(x_1, x_2)
{ y <- $x_1^2 + x_2^2$
return(y)
}
myFunction(x_1 = 2, x_2=3)

[1] 13

LA IMPORTANCIA DE TRABAJAR CON PROYECTOS

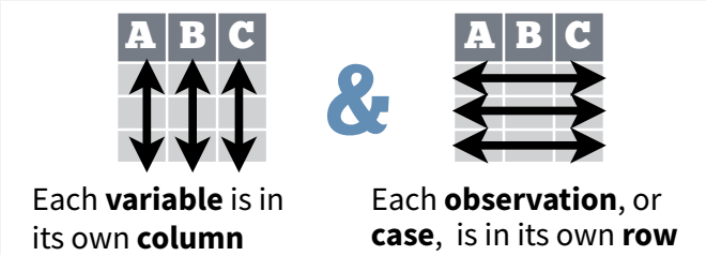
Una de las ventajas de RStudio es que permite crear "proyectos". Un proyecto es un espacio o contexto de trabajo asociado a una carpeta en particular, en la que se guardan nuestros scripts, archivos de datos, gráficos, etc. Cuando creamos un proyecto en RStudio, se crea un tipo especial de archivo (.Rproj) que lo que hace es vincular todo lo que se encuentra dentro de esa carpeta.

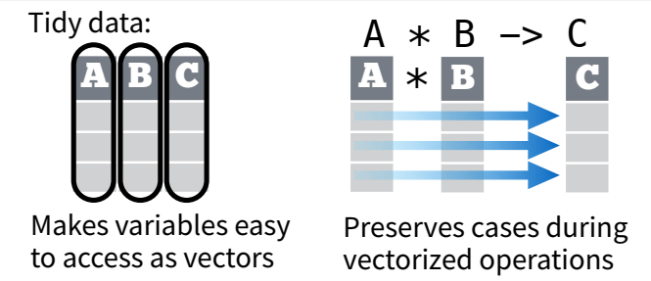
¿Por qué esto es útil? Si parte de nuestro script, por ejemplo, implica abrir un archivo que está en la carpeta de nuestro proyecto, no necesito indicar en mi código toda la ruta del archivo: lo que hará RStudio será buscarlo en el entorno/carpeta del proyecto. Si movemos la carpeta a otro lugar de nuestro computador o la compartimos con otra persona, nuestro código seguirá funcionando, ya que el archivo .Rproj mantendrá todo unido. Si no creara un proyecto, tendría que indicar al inicio de mi script cuál es la ruta de la carpeta que ocuparé como espacio de trabajo. El problema de esa opción es que si muevo la carpeta o le cambio el nombre, tendría que volver a escribir la ruta para que todo funcione. Al crear un proyecto eso deja de ser una preocupación.

IMPORTACIÓN DE DATOS EN R

Esos datos pueden ser tabulares, jerárquicos, relacionales y distribuidos.

#Tidy data: es importante tener los datos ordenados.





IMPORTACIÓN DE DATOS EN R

#Por medio de read.csv()

Iris <-read.csv("E:/DATASETS/iris.csv")

View(iris)

#Por medio de read.table ( )

mtcars <- read.table ("E:/DATASETS/mtcars.txt")

mtcars <- read.table ("E:/DATASETS/mtcars.txt", header=TRUE)

View(mtcars)

IMPORTACIÓN DE DATOS EN R

#También podemos importar otro tipo de datos mediante el paquete readxl

- **Ejemplo:**
- **install.packages ("readxl")**
- **library (readxl)**
- **estadis <- read_xlsx("estadistica2009.xlsx")**

EXPLORACIÓN BÁSICA DE DATOS CON IRIS

- El dataset más común para análisis estadísticos y ciencia de datos.
- Lo vamos a usar para ver funciones básicas para explorar un dataframe.

Iris virginica

Iris versicolor

Iris setosa

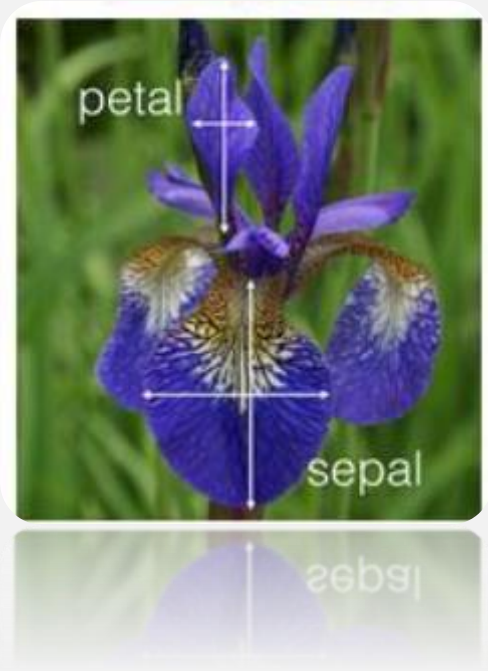
EXPLORACIÓN BÁSICA DE DATOS CON IRIS

 # Carga del dataset

data(iris)

 # Funciones básicas

- **class (iris)**
- **dim (iris)**
- **names(iris)**
- **str(iris)**
- **attributes (iris)**
- **summary (iris)**



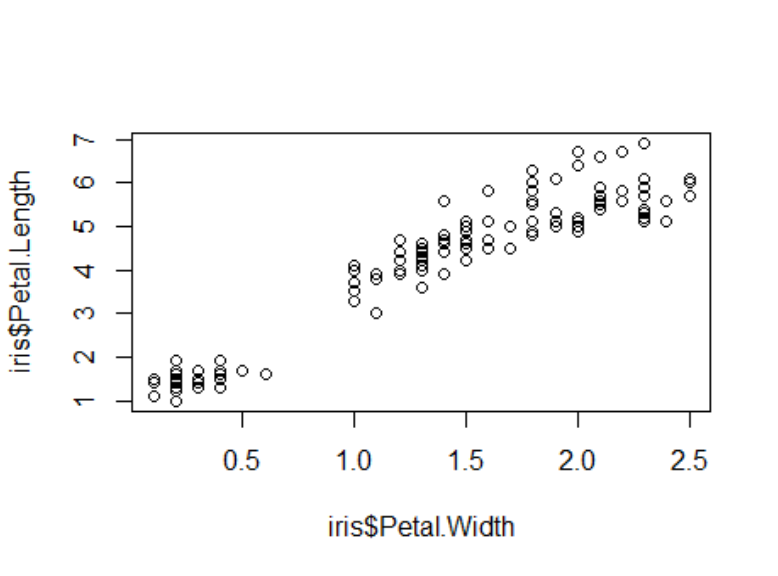
EXPLORACIÓN BÁSICA DE DATOS - IRIS

Visualización básica

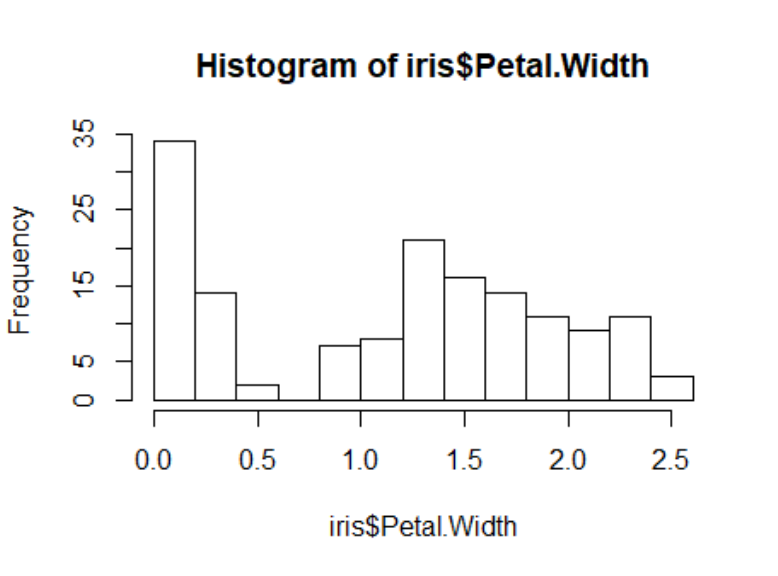
1. **plot(iris$Petal.Width, iris$Petal.Length)**
2. **hist (iris$Petal.Width)**
3. **Pie(table(iris$Species))**

EXPLORACIÓN BÁSICA DE DATOS - IRIS

1. **Plot**



2. **Histograma**



¡MUCHAS GRACIAS!

¿ESTAMOS EN CONTACTO?

Email: patricialoto@hotmail.com

Twitter: https://twitter.com/patriloto