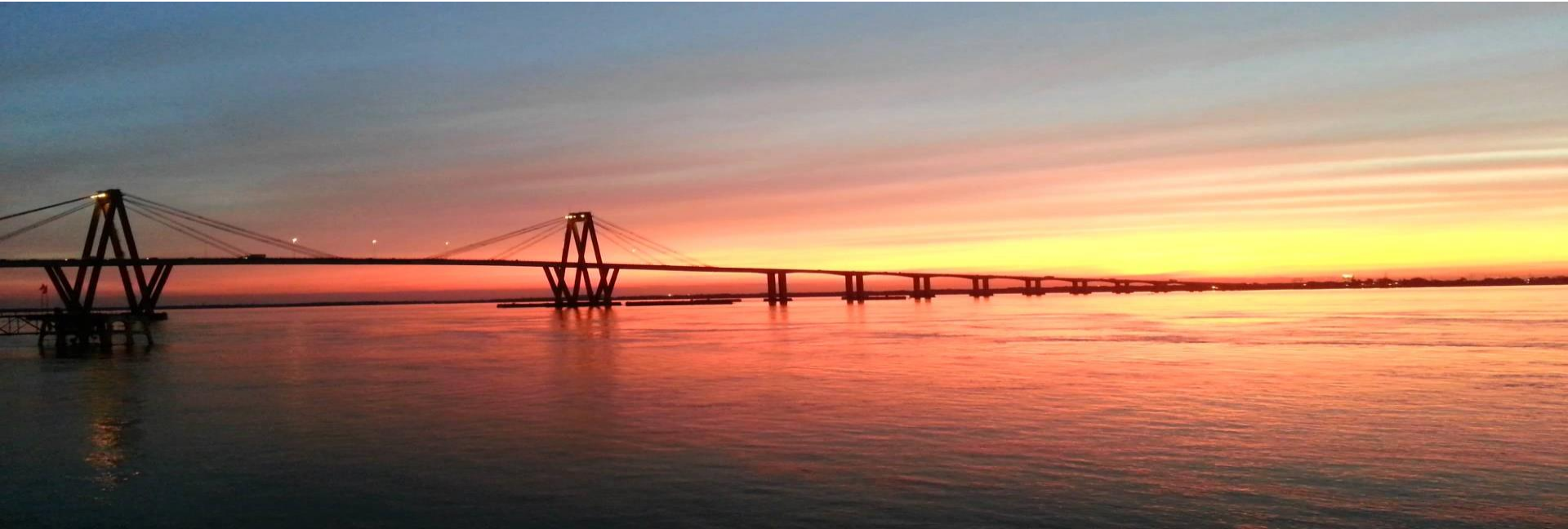
BIENVENIDOS A

R-LADIES RESISTENCIA-CORRIENTES



```
library(dplyr)
library(magrittr)

rladies_global %>% filter(city == 'Resistencia')&&  filter(city ==
'Corrientes')
```

SEGUNDO ENCUENTRO

R-LADIES RESISTENCIA-CORRIENTES

Viernes 10 de Agosto de 2018

Hoy hablamos sobre…

PARTE 1: Tipo de datos (objetos)

Datos Atómicos

Vectores

Matrices

Factores

Listas

Dataframes

Elementos de la sintaxis de R/ Coerción

PARTE 2: Estructuras de Control y Funciones

If-else

For

While

Funciones.

Limpiamos el espacio de trabajo

Importación de Datos en R

Exploración básica de datos

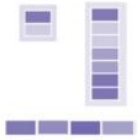
PARTE 3: Introducción a Git

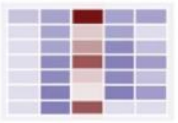
PARTE 1

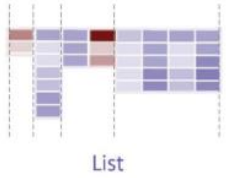
1. Tipo de objetos

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 tiene 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 ***diferentes tipos***. Puede pensarse como un ***contenedor de objetos*** que pueden ser: vectores, factores, matrices, arreglos, dataframes, 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 **x || y** idem **xor(x,y)** O exclusivo


En R todos los
datos son objetos

Datos atómicos

Pueden ser de diferentes tipos. *La función class () nos dice el tipo de dato.*

#Numéricos (reales)

```
x <- 2
class(x)
```

#Enteros

```
y <- 1L
class(x)
```

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

#Complejos

```
z <- 2+2i
class(y)
```

#Caracteres

```
xy <- "Roxana"
class(z)
```

#Lógicos o booleanos

```
yz <- TRUE
class(b)
```

Vectores

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

Se crea con la función **c()**

#Un vector de tipo numérico

```
num <- c (5,3,2)
num
```

#Un vector de tipo lógico o booleano

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

#Un vector de cadena de caracteres

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

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

```
vect <- c(num, boolean, caract)
vect
```

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
```

?seq podemos consultar la ayuda de R

#Mediante la función rep()

```
seq4 <- rep (seq3, 2)
seq4
```

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

```
length(seq3)
length(seq4)
```

Indexación

#Mediante el operador [ ]

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

Me permite acceder al segundo elemento

Matrices

Desde el punto de vista del lenguaje, una matriz es un **vector** *con atributo adicional:* ***dim***. Es un vector entero de 2 elementos, a saber: el número de renglones y el número de columnas. Un array o arreglo es lo mismo pero tiene más de 2 dimensiones.

#Por medio de la función dim

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

#Por medio de la función matrix()

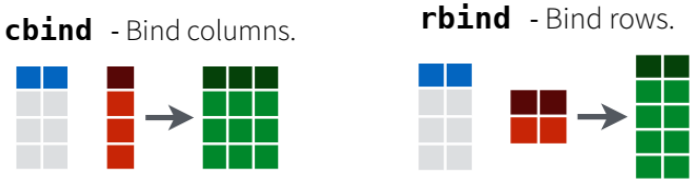
```
mat2 <- matrix(11:30, nrow=5, ncol=4)
```

#Por medio de cbind() y rbind()

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

Las matrices 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.

```
mat2 <- matrix(11:30, nrow=5, ncol=4)
mat3 <- matrix(11:30, nrow=5, ncol=4, byrow=True)
```



Matrices: Indexación

Para acceder a los elementos de una matriz

#Mediante el operador [ ]

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

m <- mat2[3,2]
k <- mat[8]
```

Se puede acceder mediante elemento de la matriz como un vector

#Mediante el operador [[ ]]

```
n <- mat2[[2]]

xy <- mat2[[2,3]]
```

`m[2, ]` - Select a row

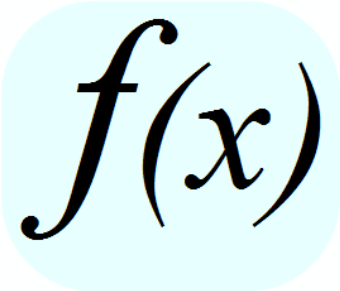
`m[ , 1]` - Select a column

`m[2, 3]` - Select an element

El operador [[ ]] a diferencia de [ ] entrega un vector de un solo elemento.

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])

print(c(persona[2], mes.nac[2]))
```

#Para concatenar factores mediante paste()

```
paste(persona[2],"nacio 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

mtcars; trees, iris, etc

#En general, los dataframes los importamos externamente 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 [[ ]]

mtcars[[1]]

#Mediante el operador $

mtcars$hp

mtcars$"drat"

mtcars$w

Sólo para listas y dataframes

Al usar **$** los nombres de las columnas no necesariamente debe ir entrecomillado, ni siquiera el nombre completo, siempre y cuando pueda ser identificado unívocamente.

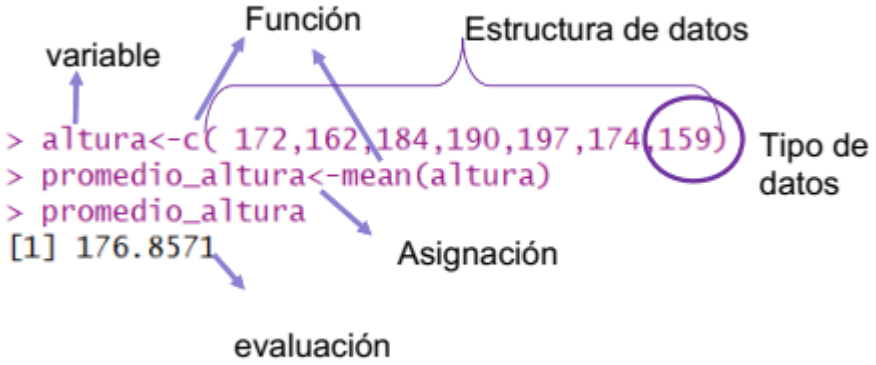
#Podemos modificar una columna

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**



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

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.

- **#repeat**

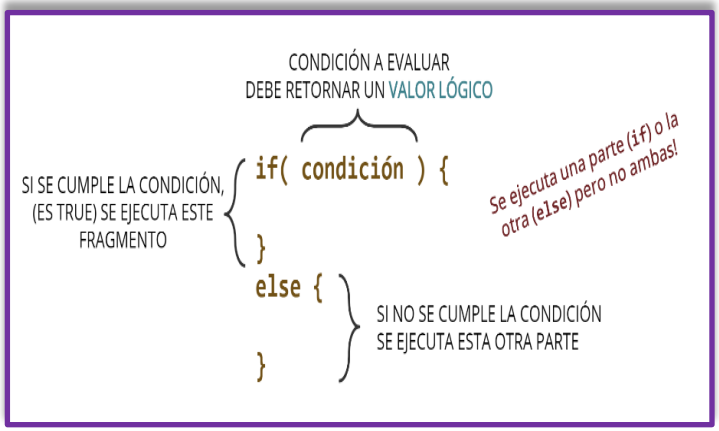
Ejecuta un bucle indefinidamente

- **#break**

Detiene la ejecución de un bucle

Estructuras de Control: if/else

Evalúa la condición, y ejecuta código dependiendo de si esa condición se cumple o no se cumple.



Ejemplo:

```
If a >b {
y ← a-b
}
else {
y ← b-a
 }
return(y)
```

Estructuras de Control: for

- Es un ciclo o bucle, el más usado de R.
- Repite ciertas líneas de código durante una cantidad conocida de veces.
- Requiere un *iterador*: una variable que toma valores sucesivos desde un origen a un destino. El iterador puede ser un vector.

```
for(<variable> in <rango>) { <instrucciones a repetir> }
```

Ejemplo:

```
for (i in 1:10)
{ x <- x + 1
}
print(x)
```

Estructuras de Control: while

- Otro tipo de bucle.
- Se repite (itera) mientras se cumpla una condición lógica, es decir, mientras la condición sea TRUE.
- En cada ciclo, la condición se vuelve a evaluar, para decidir si continuar o no ejecutando el código.

```
while(<condición>) { <instrucciones a repetir> }
```

Ejemplo:

```
suma=0
i=1
while (i < =10)
{ suma <- suma+i
   i <- i+1
}
print ("La suma es “,(suma))
```

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(x1, x2)
{ y <- x1^2 + x2^2
return(y)
}
myFunction( x1 = 2, x2=3 )

## [1] 13
```

Limpiamos el espacio de trabajo

#Mediante rm(x)
Borro solamente x.

#Para borrar todos los objetos creados
rm(list = ls ())

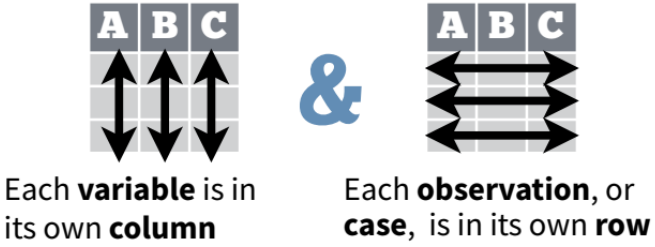
#Presionando CTRL + L borramos toda la consola de R

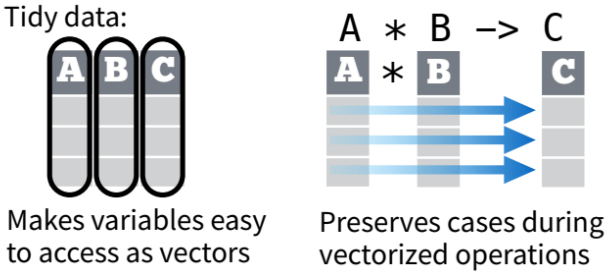
#Mediante q() cerramos R

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)

#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 - Iris

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

Iris virginica

Iris versicolor

Iris setosa

 # Ejercicio 0

Crear un archivo nuevo de R en el directorio de trabajo y llamarlo iris.R. Luego, escribir el siguiente código para explorar el dataset *iris* que trae con R base

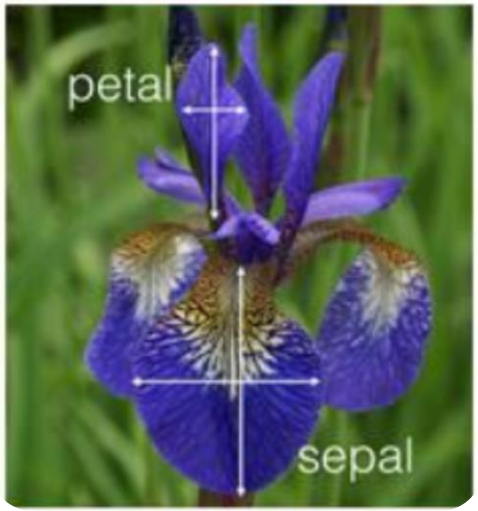
Exploración básica de datos - Iris

#Carga del dataset

data(iris)

#Funciones básicas

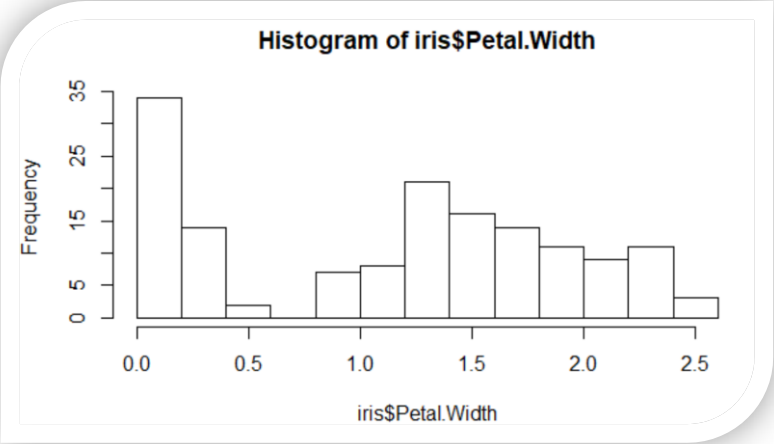
- class (iris)
- dim (iris)
- names(iris)
- str(iris)
- attributes (iris)
- summary (iris)



Exploración básica de datos - Iris

#Visualización básica

- hist (iris$Petal.Width)
- plot (iris)
- plot(iris$Petal.Width, iris$Petal.Length)
- Pie(table(iris$Species))

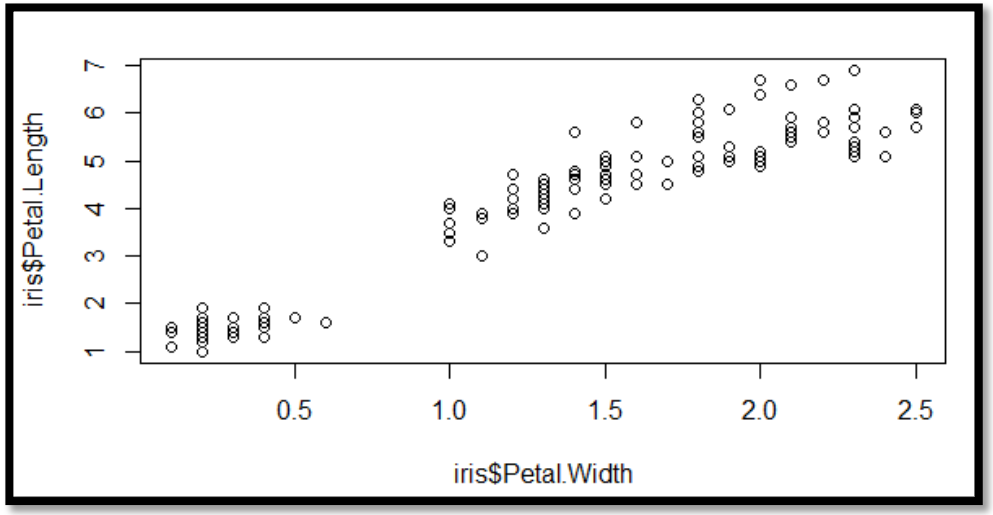


Iris virginica

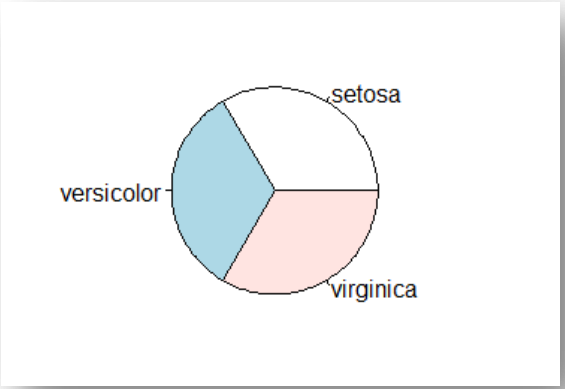
Iris setosa

Exploración básica de datos - Iris

- plot(iris$Petal.Width, iris$Petal.Length



- Pie(table(iris$Species))



CÓMO CONTACTARNOS

Email: rciactes@rladies.org

Meetup: https://www.meetup.com/rladies-resistencia-corrientes/

Twitter: https://twitter.com/RLadiesRciaCtes

Facebook: https://www.facebook.com/R-Ladies-Rcia-Ctes-1959095607753886/

Slack: https://rladies-rcia-ctes.slack.com/