

# Ciencias de Datos con R:

## Fundamentos Estadísticos

Mariela Sued, Jemina García y Ana M. Bianco

26 marzo 2020

# ¿Cómo nos vamos a organizar?

- ▶ Video
- ▶ pdf con los slides
- ▶ R abierto
- ▶ Script de la clase
- ▶ Guía de ejercicios
- ▶ Script de Guía

# Bibliografía

- ▶ General

[https://mathstat.slu.edu/~speegle/\\_book/preface.html#installing-](https://mathstat.slu.edu/~speegle/_book/preface.html#installing-)

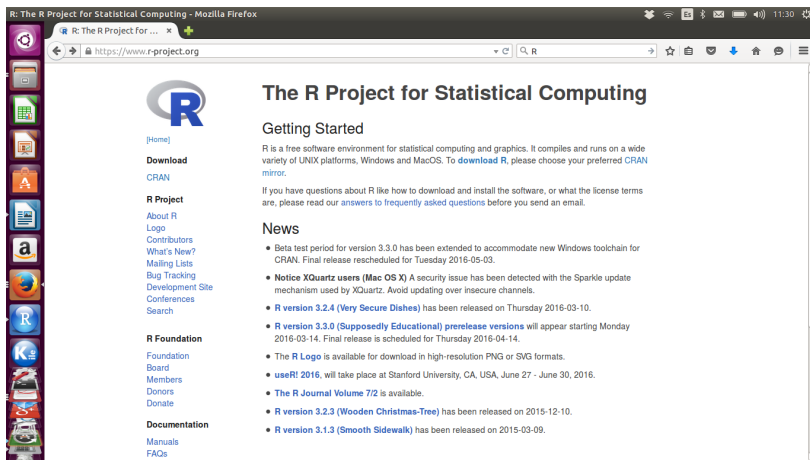
- ▶ Para gráficos (entre muchas otras...)

<https://www.statmethods.net/advgraphs/parameters.html>

# ¿Qué es R?

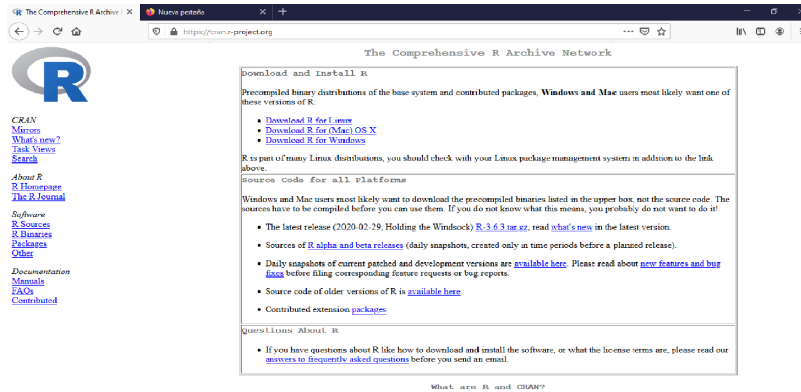
- ▶ R es un lenguaje de programación.
- ▶ Fue diseñado para el análisis de datos y la elaboración de gráficos.
- ▶ Software libre, corre en diferentes sistemas operativos.
- ▶ Interacción por línea de comandos (reglas de sintaxis).
- ▶ <https://www.r-project.org/>
- ▶ <https://cran.r-project.org/>

# Página de R



Figure

# Página de R



The screenshot shows a web browser window with the address bar displaying "https://cran.r-project.org". The page title is "The Comprehensive R Archive Network". On the left side, there is a navigation menu with links: "CRAN", "Mirrors", "What's new?", "Task Views", "Search", "About R", "R Homepage", "The R Journal", "Software", "R Sources", "R Binaries", "Packages", "Other", "Documentation", "Manuals", "FAQs", and "Contributed". The main content area is titled "Download and Install R" and contains the following text: "Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:" followed by a bulleted list of links: "Download R for Linux", "Download R for (Mac) OS X", and "Download R for Windows". Below this, it states: "R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above." and "Source code for all platforms". Further down, it says: "Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!" followed by a bulleted list: "The latest release (2020-02-29, Holding the Windsock) [R 3.6.3 tar.gz](#), read [what's new](#) in the latest version.", "Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).", "Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.", "Source code of older versions of R is [available here](#)", and "Contributed extension [packages](#)". At the bottom, there is a section titled "Questions About R" with a bulleted list: "If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email." The footer of the page says "What are R and CRAN?".

The Comprehensive R Archive Network

## Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source code for all platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2020-02-29, Holding the Windsock) [R 3.6.3 tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#)
- Contributed extension [packages](#)

## Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

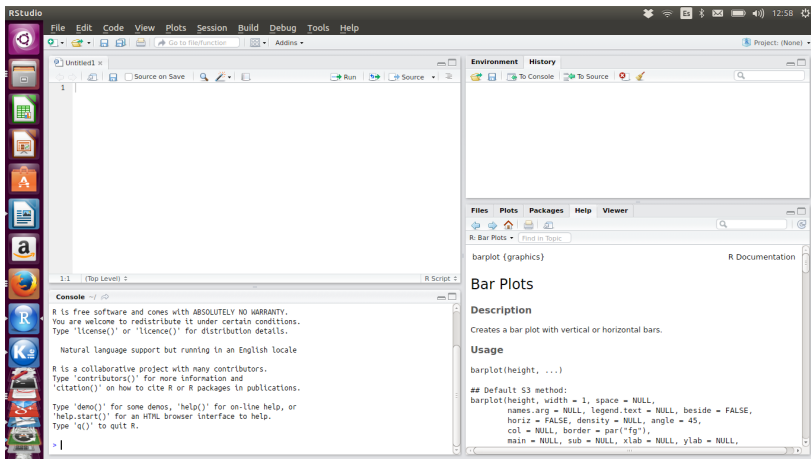
What are R and CRAN?

Figure

# Rstudio ¿Qué es?

- ▶ Es una interfase gráfica a R.
- ▶ Es un entorno *amigable* donde ejecutar R.
- ▶ <https://www.rstudio.com/>

# Pantalla Rstudio



Figure



# Pantalla de Rstudio

- ▶ Console: Ejecuta comandos y muestra los resultados.
- ▶ Editor: Acá se escribe lo que se quiere ejecutar (script)
- ▶ History - Environment
- ▶ Files - Plots - Packages - Help - Viewer

# Instalación de paquetes

Vamos a usar más adelante un paquete para realizar gráficos y hay que instalarlo:

Por comando:

- ▶ `install.packages("ggplot2")`

Por pantalla:

- ▶ usando la solapa Tools

## R como calculadora

- ▶  $2+7$ ,  $5-3$
- ▶  $9*3$ ,  $62/3$
- ▶ potencia :  $2^3$
- ▶ `sqrt(4)`

## R como calculadora

- ▶  $2+7$ ,  $5-3$
- ▶  $9*3$ ,  $62/3$
- ▶ potencia :  $2^3$
- ▶ `sqrt(4)`
  
- ▶  $\pi*2$
- ▶ `exp(1)`
- ▶ `log(8)` o con otra base `log(8, base = 2)`
- ▶ `cos(pi)`

## Asignación - Creación de objetos

Podemos darle nombre a las cosas y asociarles un valor. Esto se llama asignar un valor a una variable:

`< - =`

- ▶ Se consigue `< -` con el menor, seguido del guión o con **Alt +-** que es el shortcut o atajo en la línea de comandos.
- ▶ `pepe< - 3`: crea el objeto pepe y le asigna el valor 3.
- ▶ `nombre< - "Pirulo"`: crea un objeto de tipo string y le asigna el valor Pirulo

# Tipos Simples

Ejemplos de asignación de variables.

- ▶ `peso<- 3.2`
- ▶ `nombre <- "Jose"`

**Ojo:** distingue mayúsculas de minúsculas, no es la misma variable nombre que Nombre.

Otras maneras para efectuar asignaciones.

- ▶ `nombre2 = "Agustin"`

Consultar todos los objetos cargados en el entorno.

- ▶ `objects()` o bien `ls()`

# Operaciones

```
peso<- 3.2  
peso*1000 #Salida directa en pantalla - Esto es un comentario.
```

```
## [1] 3200
```

# Operaciones

```
peso<- 3.2  
peso*1000 #Salida directa en pantalla - Esto es un comentario.
```

```
## [1] 3200
```

```
peso<- 3.2           #peso en kilos - Esto es un comentario.  
peso <- peso*1000  
peso
```

```
## [1] 3200
```

- ▶ ¿Qué diferencia hay?
- ▶ *rm* borra objetos: `rm(peso)`



## Comentarios:

```
peso<- 3.2  
peso*1000 #Salida en pantalla - Esto es un comentario.
```

```
## [1] 3200
```

*Comentarios:* son provistos para darle al lector información sobre lo que está ocurriendo en el código R, pero ni son ejecutados ni tienen ningún impacto sobre el resultado.

# Vectores

- ▶ `c(1,2,4)`
- ▶ `c`: concatena
- ▶ `prueba.vec <- c(1,2,4)`: crea el objeto `prueba.vec` y le asigna los valores 1,2 y 4.
- ▶ `x <- c(2,4,6,8,10)`
- ▶ `y <- 1:5` *Otra forma de generar vectores*

# Vectores

- ▶ `c(1,2,4)`
- ▶ `c`: concatena
- ▶ `prueba.vec <- c(1,2,4)`: crea el objeto `prueba.vec` y le asigna los valores 1,2 y 4.
- ▶ `x <- c(2,4,6,8,10)`
- ▶ `y <- 1:5` *Otra forma de generar vectores*

## Comando `rep` - `rep(x, times)`

- ▶ `x <- rep(0,10)`
- ▶ `rep(1:3,4)` *Repetimos varias veces una lista.*
- ▶ `x <- rep(NA,5)` **NA: Not Available, valor ausente, missing**

# Vectores

- ▶ `nombres <- c("Jose", "Pedro", "Agustin")`
- ▶ ¿Qué diferencia hay?
- ▶ `rep(c("Jose", "Pedro"), times = 2)`
- ▶ `rep(c("Jose", "Pedro"), each = 2)`

# Vectores

- ▶ `nombres <- c("Jose", "Pedro", "Agustin")`
- ▶ ¿Qué diferencia hay?
- ▶ `rep(c("Jose", "Pedro"), times = 2)`
- ▶ `rep(c("Jose", "Pedro"), each = 2)`

## help - help()

- ▶ Cuando queremos saber algo sobre un comando usamos `help`

`help(rep)`

## Vectores: Comando seq - seq(from=a, to=b, by=c)

- ▶ `r <- seq(1,5,0.5)` Especificamos un 'step' o 'by'

```
seq(1,5,0.5)
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

- ▶ `r <- seq(1,-3,-0.5)` Podemor también ir para atrás. Con un paso de 0.5 positivo obtendríamos un error.
- ▶ `seq(-5,5, length=10)` Nos armamos una tira de 10 elementos.
- ▶ `w <- seq(1,5)` Equivalente a `1:5`

*Notar la manera especial con la cual podemos pasar los parámetros a las funciones.*

*Además, si no se asigna el resultado a una variable sale lo obtenido por pantalla.*

## Operaciones con un Vector

```
y<- seq(1,7,by=2)
```

```
y
```

```
## [1] 1 3 5 7
```

```
2 * y
```

```
## [1] 2 6 10 14
```

```
5+y
```

```
## [1] 6 8 10 12
```

```
log(y)
```

```
## [1] 0.000000 1.098612 1.609438 1.945910
```

# Operaciones entre Vectores

*Podemos operar directamente sobre vectores*

```
x <- 1:4
```

```
x + y
```

```
## [1] 2 5 8 11
```

```
x*y
```

```
## [1] 1 6 15 28
```

► `w <- x * y` ¿Qué hace?



## Comandos sobre vectores

```
sum(c(2,4,6))
```

```
## [1] 12
```

► Sin embargo:

```
sum(c(2,4,NA))
```

```
## [1] NA
```

## Comandos sobre vectores

```
sum(c(2,4,6))
```

```
## [1] 12
```

► Sin embargo:

```
sum(c(2,4,NA))
```

```
## [1] NA
```

```
sum(c(2,4,NA),na.rm =T)
```

```
## [1] 6
```

# Indexación de Vectores

```
w<- (1:4)*10      #Los arreglos se indexan desde el 1  
w[2]
```

```
## [1] 20
```

```
v <- rep(0,5)  
v[c(1,4,5)] <- 1  
v
```

```
## [1] 1 0 0 1 1
```

```
v[1:2]
```

```
## [1] 1 0
```

```
w[-4]
```

```
## [1] 10 20 30
```

## Vectores lógicos.

```
x <- c(2,4,7,20)
y <- c(3,2,8,19)
x>y
```

```
## [1] FALSE TRUE FALSE TRUE
```

```
z <- x>y #z es un vector con valores lógicos.
```

```
z
```

```
## [1] FALSE TRUE FALSE TRUE
```

```
sum(z)
```

```
## [1] 2
```

```
mean(x>5) #que hace esto?
```

```
## [1] 0.5
```

# Indexación de Vectores: un poco más

```
x
```

```
## [1] 2 4 7 20
```

```
x>6
```

```
## [1] FALSE FALSE TRUE TRUE
```

```
x[x>6] # elige las componentes que cumplen la condición
```

```
## [1] 7 20
```

```
x[c(F,F,T,T)] # También se podría hacer así
```

```
## [1] 7 20
```

# Comparaciones

- ▶  $<$  menor
- ▶  $<=$  menor o igual
- ▶  $>$  mayor
- ▶  $>=$  mayor o igual
- ▶  $==$  igual
- ▶  $!=$  diferente

## Un poquito más...

Buscando el máximo y el mínimo valor del vector

```
y<- c(10,30,15,5)  
y
```

```
## [1] 10 30 15 5
```

```
max(y)
```

```
## [1] 30
```

```
min(y)
```

```
## [1] 5
```

## Un poquito más...

```
y
```

```
## [1] 10 30 15 5
```

```
which.max(y)
```

```
## [1] 2
```

```
which.min(y)
```

```
## [1] 4
```

```
sort(y)
```

```
## [1] 5 10 15 30
```

```
order(y)
```



## Vectores: breve resumen

Comando	Acción
<code>c(a,b,c)</code>	crea vector concatenando $a, b, c$
<code>rep(a,n)</code>	repite $a$ $n$ -veces
<code>seq(1:n)</code>	equivale a $1:n$ & $c(1, 2, 3, \dots, n)$
<code>seq(a,b,by=c)</code>	$(a, a + c, a + 2c, \dots)$ hasta $b$
<code>x[4]</code>	elige la cuarta coordenada del vector $x$
<code>x[c(2,5)]</code>	elige la segunda y la quinta coordenadas del vector $x$
<code>length(x)</code>	calcula la longitud del vector $x$
<code>which.max(x)</code>	determina la posición del máximo valor de $x$
<code>which.min(x)</code>	determina la posición del mínimo valor de $x$

## Cosas útiles

Comando	Acción
<code>sum(vec == 3)</code>	número de veces que 3 aparece en vec
<code>mean(vec == 3)</code>	proporción de veces que 3 aparece en vec
<code>table(vec)</code>	número de veces que cada valor ocurre en vec
<code>max(table(vec))</code>	valor que más veces ocurre en vec
<code>length(unique(vec))</code>	número de valores distintos que ocurren en vec
<code>vec[vec &gt; 0]</code>	consta solo de los valores positivos
<code>vec[!is.na(vec)]</code>	consta solo de los valores que no son missing

Notar que `vec[vec > 0]` | y `vec[!is.na(vec)]` son nuevos vectores.

## Area y directorio de trabajo

- ▶ Cuando comenzamos es aconsejable borrar todos los objetos existentes en el entorno de trabajo y establecer el directorio de trabajo. Para borrar todo:

```
rm(list=ls())
```

- ▶ Para saber en que directorio estamos, hacer *getwd*
- ▶ Para cambiarlo: *setwd*

```
setwd("C:/Users/Ana/Dropbox/Ciencias_de  
Datos_Fundamentos/optativa_2020/Clase_1")
```

# Data Frames

- ▶ Se utilizan para guardar los ficheros de datos
- ▶ Intuitivamente son matrices con entradas de distintos tipos, vectores (heterogéneos) de la misma longitud
- ▶ Para crearlos se utiliza el comando *data.frame*

```
x <- 7:9
y <- c('F','M','F')
misdatos <- data.frame(edad=x, sexo=y)
misdatos
```

```
##      edad sexo
## 1      7    F
## 2      8    M
## 3      9    F
```

# Data Frames

Admiten comandos para matrices y para listas

```
misdatos$edad
```

```
## [1] 7 8 9
```

```
misdatos[2,]
```

```
##    edad sexo
```

```
## 2     8    M
```

# Importar datos a R

Desde el directorio de trabajo que fije anteriormente

```
datos<- read.table("iris.txt",sep=" ",dec=".",header=T)
```

---

## Argumento

---

sep	caracter que se usa para separar variables
dec	caracter que separa la parte entera y la decimal de cada numero
header	es TRUE si la primera linea contiene el nombres de las variables

---

# Gráficos

Una herramienta que usaremos mucho para comunicar resultados son los gráficos. Veremos en esta instancia los más sencillos.

\* Tenemos una tabla de valores. \* Graficamos cada punto, empezando con el (1, 1), agregar el (2, 4), y así seguimos. . .

► Vemos la necesidad de automatizar:

```
xx<-c(1, 2, 3, 4, 5, 6, 7)
```

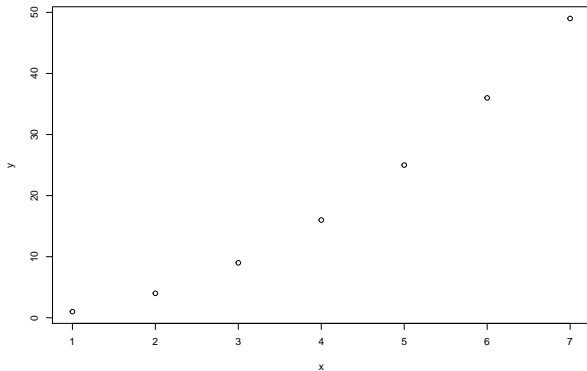
```
yy<-c(1, 4, 9, 16, 25, 36, 49)
```

# Gráficos

```
x<-c(1,2,3,4,5,6,7)
```

```
y<- c(1,4,9,16,25,36,49)
```

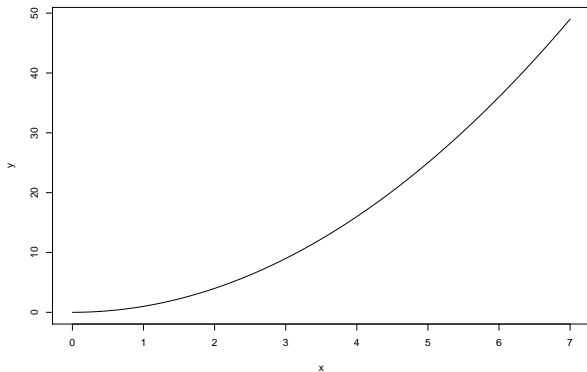
```
plot(x,y)
```





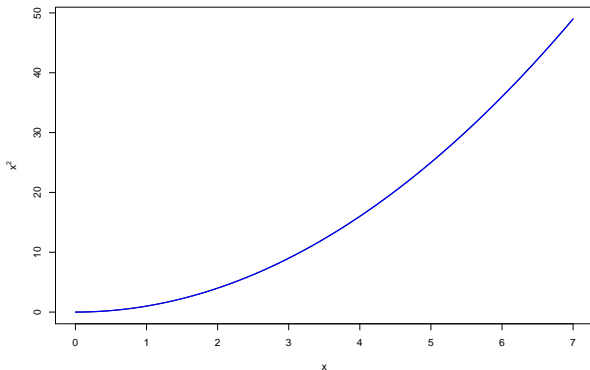
# Gráfico Continuo

```
x<-seq(0,7,by=0.01)  
y<- x*x  
plot(x,y,type="l")
```



# Gráfico Continuo

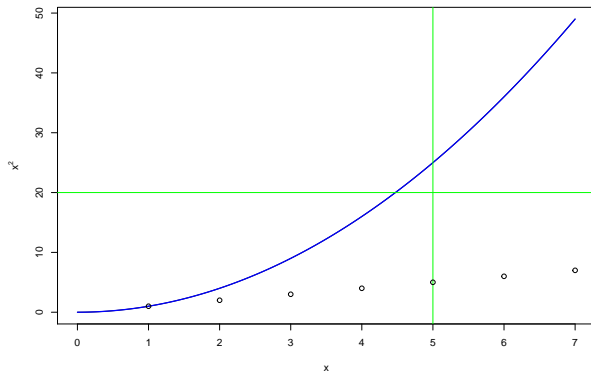
```
x<-seq(0,7,by=0.01)
y<- x*x
plot(x,y,type="l",ylab=expression(x^2))
lines(x,y,col="blue",lwd=2)
```



# Gráfico Continuo

Podemos agregar algunas cosas más, tales como rectas o puntos

```
plot(x,y,type="l",ylab=expression(x^2))  
lines(x,y,col="blue",lwd=2)  
abline(h=20, v=5,col="green")  
points(1:7)
```



# Datos

Estos datos también están cargados en R.

```
data(iris)
#Inspecciono los primeros y los ultimos casos.
head(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

```
tail(iris)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 145	6.7	3.3	5.7	2.5	virginica
## 146	6.7	3.0	5.2	2.3	virginica
## 147	6.3	2.5	5.0	1.9	virginica
## 148	6.5	3.0	5.2	2.0	virginica
## 149	6.2	3.4	5.4	2.3	virginica
## 150	5.9	3.0	5.1	1.8	virginica

# Datos

Si ejecuto

```
fix(iris)
```

abre los datos

El attach de iris guarda los nombres de las variables

```
attach(iris)
```

Muestra los data.frames y paquetes atachados

```
search()
```

```
## [1] ".GlobalEnv"          "iris"                "package:stats"
## [4] "package:graphics"    "package:grDevices"   "package:utils"
## [7] "package:datasets"    "package:methods"     "Autoloads"
## [10] "package:base"
```

# Inspeccione los nombres y el tipo de las variables de datos

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  
## [5] "Species"
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:  
## $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
```

```
dim(iris)
```

```
## [1] 150    5
```

# Práctica

Vamos a resolver la guía.

# Operadores

Aritméticos		Comparativos	
+	Adición	==	Igual a
-	Substracción	!=	Diferente de
*/	Multiplicación División	<, >	Menor que, Mayor que



# Funciones

Funciones matemáticas		Funciones estadísticas	
<code>sqrt(x)</code>	Raíz de x	<code>mean(x)</code>	Media
<code>exp(x)</code>	Exponencial de x	<code>sd(x)</code>	Desvio
<code>log(x)</code>	Logaritmo natural de x	<code>var(x)</code>	Varianza
<code>log10(x)</code>	Logaritmo base 10	<code>median(x)</code>	Mediana
<code>length(x)</code>	Número de elementos	<code>quantile(x,p)</code>	Quantiles
<code>sum(x)</code>	Suma los elementos de x	<code>max(x)</code>	El máximo
<code>prod(x)</code>	Producto de los elementos	<code>min(x)</code>	El mínimo
<code>sin(x)</code>	Seno	<code>summary(x)</code>	Resumen
<code>cos(x)</code>	Coseno	<code>sort(x)</code>	Ordena (creciente)
<code>tan(x)</code>	Tangente		
<code>round(x,n)</code>	redondea a n dígitos		
<code>cumsum(x)</code>	calcula las sumas acumuladas		
<code>choose(n, k)</code>	calcula en combinatorio		