

---

```
library(dplyr)
```

```
rladies_global %>%  
  filter(city == 'Córdoba')
```



# Taller introducción a R



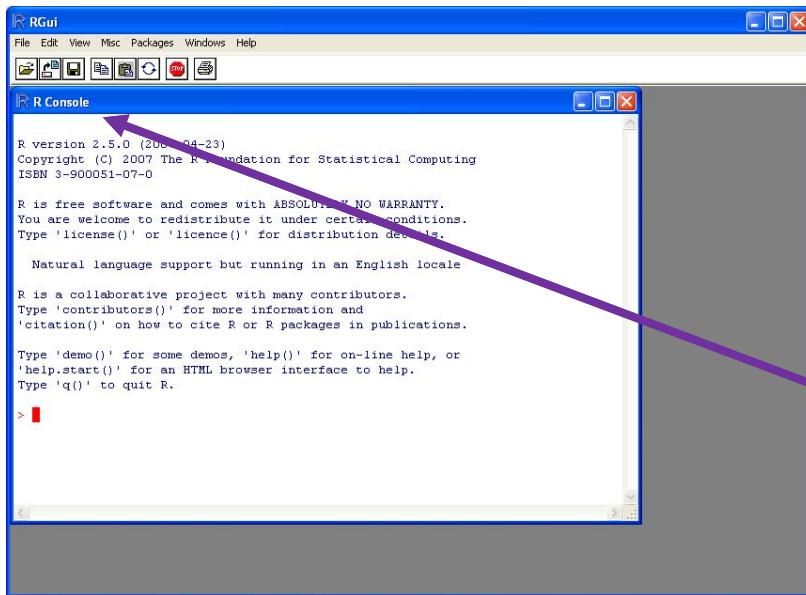


# R y RStudio

RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R



# R

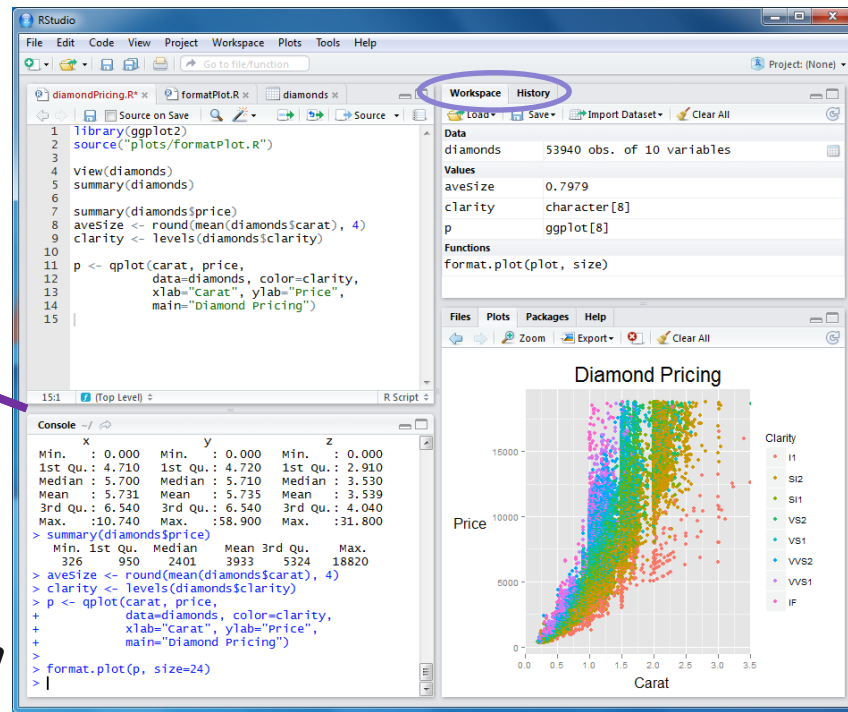


*Consola*

# RStudio

*Fuente*

*Entorno e historia*



*Archivos, gráficos, paquetes y ayuda*

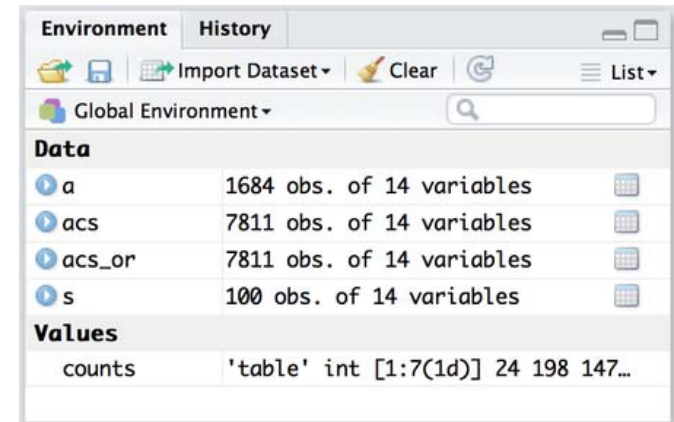
# Entorno e historia

## ▪Environment

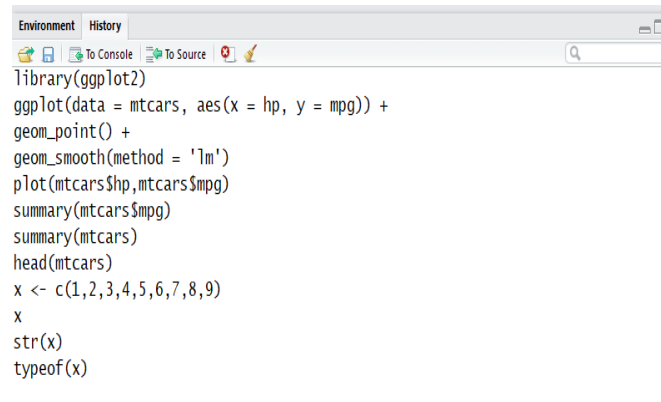
Pueden verse los objetos creados con descripción de su estructura, valores, etc.

## History

▪lista de todos los comandos ejecutados durante la sesión o proyecto actual



Environment		History
Global Environment		
<b>Data</b>		
a	1684 obs. of 14 variables	
acs	7811 obs. of 14 variables	
acs_or	7811 obs. of 14 variables	
s	100 obs. of 14 variables	
<b>Values</b>		
counts	'table' int [1:7(1d)] 24 198 147...	



```
library(ggplot2)
ggplot(data = mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth(method = 'lm')
plot(mtcars$hp,mtcars$mpg)
summary(mtcars$mpg)
summary(mtcars)
head(mtcars)
x <- c(1,2,3,4,5,6,7,8,9)
x
str(x)
typeof(x)
```

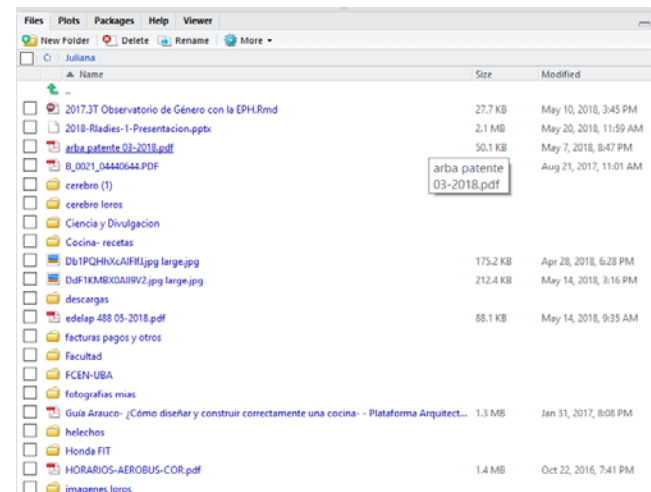
# Archivos, gráficos, paquetes y ayuda

## ■ Archivos

■ La pestaña *Files* nos muestra los archivos guardados en el *wd* y también gestionar archivos o navegar por otros directorios sin tener que salir de RStudio.

## ■ Plots

■ La pestaña *Plots* muestra los gráficos que resultan de usar distintas funciones que generan gráficos.



# Paquetes

funciones, datos y código compilado de R  
en un formato definido



Instalar paquete

`install.packages("tidyverse")`

**Se instala solo una vez**

O también via: tools → install packages

Cargar paquete

`library(tidyverse)`

**Se carga en cada nueva sesión**

Name	Description	Version
<input type="checkbox"/> exprm	matrix expression, Log, etc.	0.3.0-2
<input type="checkbox"/> extrafont	Tools for using fonts	0.17
<input type="checkbox"/> extrafontdb	Package for holding the database for the extrafont package	1.0
<input type="checkbox"/> fastmatch	Fast match() function	1.1-0
<input type="checkbox"/> fitdistrplus	Help to Fit of a Parametric Distribution to Non-Censored or Censored Data	1.0-9
<input type="checkbox"/> FNN	Fast Nearest Neighbor Search Algorithms and Applications	1.1
<input type="checkbox"/> forcats	Tools for Working with Categorical Variables (Factors)	0.3.0
<input type="checkbox"/> formatR	Format R Code Automatically	1.5
<input type="checkbox"/> Formula	Extended Model Formulas	1.2-2
<input type="checkbox"/> Fragman	Fragment Analysis in R	1.0.7
<input type="checkbox"/> gdata	Various R Programming Tools for Data Manipulation	2.18.0
<input type="checkbox"/> genepop	Population Genetic Data Analysis Using Genepop	1.0.5
<input type="checkbox"/> geometry	Mesh Generation and Surface Tessellation	0.3-6
<input checked="" type="checkbox"/> ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics	2.2.1
<input type="checkbox"/> ggrepel	Repulsive Text and Label Geoms for 'ggplot2'	0.7.0

Todas las funciones y operadores básicos de R están cargados automáticamente en el paquete base.  
Hay otras funciones que se cargan agrupadas en otros paquetes incluidos en el inicio (están en la parte de system library) y por otra parte está la user library

---

# Consola



Donde se ejecutan las instrucciones y se ven las salidas

- Simbolo de asignación : < - (Alt-)

Flechas :

**Arriba** — Recupera comandos anteriores

**Abajo** — Reversa de Arriba

**Ctrl+Arriba** -- Recupera la lista de los ultimos comandos y permite recorrerla para seleccionar alguno

## El directorio de trabajo (wd)

Es el ambiente de trabajo actual, donde se ejecutan las instrucciones, se van guardando todos los objetos que vas creando en una sesión de trabajo

O sino en el menú **Session, Set Working Directory, Choose Directory**



# Ayuda



Dentro de Rstudio

?Función

help(función)

F1 mientras se tipea la función

Pestaña help

En Internet:



Nombre de la función + R

Copiar y pegar el error

- Mirar si existe una pregunta similar en Stack Overflow



---

# Fuente



Donde se escribe el código para guardarlo, modificarlo y volver a usar

## Buenos hábitos:

Tener al listadas al principio todos los paquetes o librerías que se necesitan para correr este tipo de datos o análisis

Comentar el código : # todo lo que este después del numeral no lo corre el #programa, comentar el código es útil, si se vuelve a abrir un archivo meses #después, no te estas preguntando para que servía o que hacia esa línea de #código

## Recordar!!!

No empezar el nombre de una variable con un punto o un número

Los nombres de las variables son sensibles a las mayúsculas



---

## R puede ser usado como una calculadora



Se puede tipear una expresión y apretar enter o click en el botón Run (en Rstudio).

\* para multiplicación.

^ elevar a la potencia de.

El orden de las operaciones aritméticas es:

(izquierda [se computa primero] a derecha [se computa último]): ^ / \* - +





Si al resultado lo asignamos a una variable, queda “almacenado”:

```
a <- 1
```

Tipeando el nombre de la variable se obtiene el resultado

```
A
```

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

Puede almacenarse un dato/ variable y cambiarle el valor reescribiendo

```
b <- 2
```

```
b <- exp(b)
```

---

# Tipos de datos



R soporta 5 tipos básicos de datos:

Integer (enteros) (2L)

Numeric (reales) (22.3)

Character (alfanúmericos letras y números) ('a')

Logical (lógicos) (TRUE/FALSE)

Complex (números complejos) ( $3 + 2i$ )

Los valores faltantes: NA (not available)

valores numéricos no-finitos como  $\pm \infty$  con Inf y -Inf, o valores que no son numéricos con NaN ('not a number')

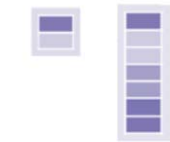
**Coerción implícita:** R interpreta automáticamente qué tipo de datos son cuando lee datos de tipo character, numeric o logical. Si lee un dato y no lo puede asignar, pone NA y avisa con un *warning*.

**Coerción explícita:** se le indica a R que interprete el tipo de datos de una manera determinada

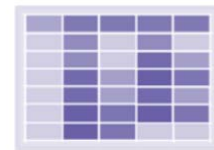


# Objetos

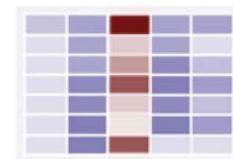
- **Vector** secuencia de elementos de datos del mismo tipo. función `c()`.
- **matrix** es una tabla de 2 dimensiones en que cada elemento es del mismo tipo. función `matrix()`.
- **Array** es similar a matrix pero puede tener mas de 2 dimensiones. función `array()`.
- **Data frame** es una matrix en la que cada columna puede ser de diferente tipo. función `data.frame()`.
- **lista** es como un data frame pero cada columna puede tener diferente longitud u objeto de los tipos anteriores. función `list()`.



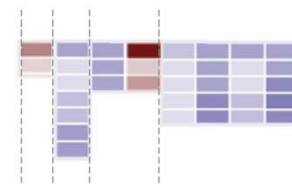
Vector



Matrix



Data.frame



List





## Como crear vectores

función c() concatena valores

```
a<-c(6,9,12)
b<-c(TRUE,TRUE,FALSE)
c<-c("manzana","naranja", "limon")
```

el operador : permite generar secuencias

```
n<-1:15

n
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

función rep() permite repetir secuencias de valores

```
r<-rep(1:5,2)

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





## Como crear matrices

Las matrices se construyen **columna a columna**, empezando por la fila superior izquierda, completando las columnas para abajo hasta alcanzar la dimensión de filas, y luego sigue agregando las columnas siguientes de a una por vez hacia la derecha.

```
> matrix1<-matrix(nrow=3,ncol = 4)
```

```
> matrix1
```

```
      [,1] [,2] [,3] [,4]  
[1,]    NA    NA    NA    NA  
[2,]    NA    NA    NA    NA  
[3,]    NA    NA    NA    NA
```

```
> matrix2<-matrix(1:12,nrow=3,ncol = 4)
```

```
> matrix2
```

```
      [,1] [,2] [,3] [,4]  
[1,]     1     4     7    10  
[2,]     2     5     8     9  
[3,]     3     6     9    12
```

También se puede crear una matriz directamente desde un vector, agregándole un atributo de dimensión.

```
n<-1:15
```



```
> dim(n)<-c(3,5)
```

```
> n
```

```
      [,1] [,2] [,3] [,4] [,5]  
[1,]     1     4     7    10    13  
[2,]     2     5     8     9    14  
[3,]     3     6     9    12    15
```

```
· n
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
str(matrix1)
```

```
logi [1:3, 1:4] NA NA NA NA NA NA ...
```

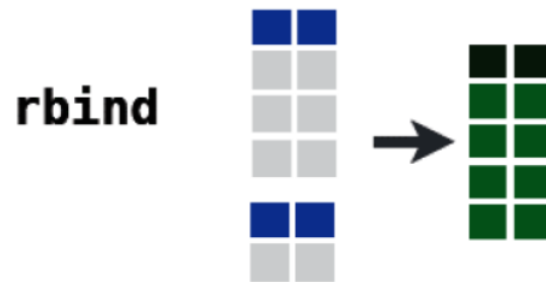
```
str(matrix2)
```

```
int  [1:3, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
```

# Como crear matrices



Pueden crearse agrupando por columnas o filas con las funciones `rbind()` y `cbind()`



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

```
> cbind(a,d)  
      a  d  
[1,]  6  7  
[2,]  9 10  
[3,] 12 13
```

```
> rbind(a,d)  
[,1] [,2] [,3]  
a    6    9   12  
d    7   10   13  
.
```





## Como crear data frame

`data.frame()` toma vectores con nombres como datos de entradas Tiene dos dimensiones como las matrices

Todas las **columnas** tienen el **mismo número de filas**

Cada **columna** tiene **datos del mismo tipo**

Pueden almacenar **distinto tipo de datos en cada columna**

```
x<-data.frame(sitios =1:5,muestreado= c(TRUE,TRUE,TRUE,FALSE,FALSE))
```

- Es un caso especial de lista.
- Es una lista de vectores que tienen todos la misma longitud.

La forma mas común de crear un df es leyendo un archivo, ej. usando las funciones `read.table()` o `read.csv()`.

Por defecto la creacion de data.frames convierte los datos de texto en factors. Hay que indicar *stringsAsFactors = FALSE* para evitar esto

```
> x
  sitios muestreado
1      1        TRUE
2      2        TRUE
3      3        TRUE
4      4       FALSE
5      5       FALSE
```



# Factor

Un factor es un vector utilizado que almacena datos categóricos para especificar una clasificación de los elementos de otro vector de igual longitud

Puede ser atómico (está solo) o ser una columna de un dataframe; puede tener etiquetas (label) que denominan a cada valor del factor y lo auto-describen, un factor tiene atributos de nivel (level) y clase "factor".

Los niveles de los factores se almacenan en orden alfabético, o en el orden en que se especificaron en la función Factor() si se hizo explícitamente.

Los factores son importantes en modelos estadísticos ya que definen los estratos o grupos a comparar o diferenciar y se usan especialmente en funciones que generan modelos como lm() y glm()

```
> x <- factor(c("Hombre", "Mujer", "Hombre", "Hombre", "Mujer"))
>x
[1] Hombre Mujer  Hombre Hombre Mujer
Levels: Hombre Mujer

## tabla de frecuencia simple
>table(x)
x
Hombre  Mujer
      3      2
```



## R Base

- Difícil de aprender al comienzo
- Permite entender programación en profundidad
- Estable

```
> x <- 1:20  
> sum(log(sqrt(x)))  
[1] 21.16781
```

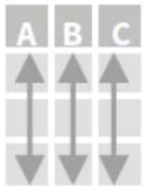
## R Moderno (data.table, tidyverse, etc)

- Más fácil de aprender
- Facilita el trabajo con datos
- Inestable y en constante cambio

```
> x <- 1:20  
> x %>% sqrt() %>% log() %>% sum()  
[1] 21.16781
```



**dplyr** funciona con conductos y requiere **datos ordenados**.  
En datos ordenados:



Cada **variable** tiene su propia **columna**

&



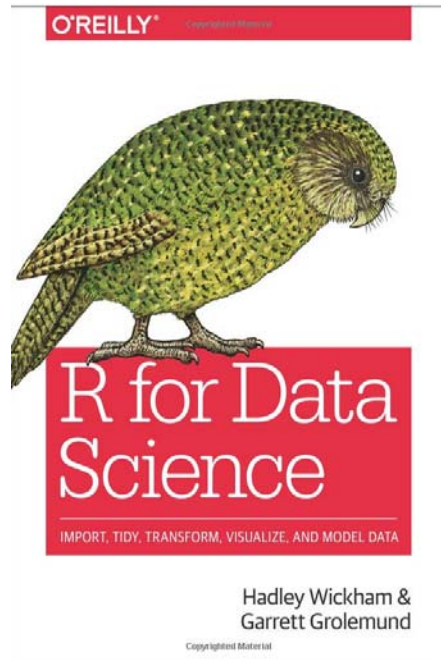
Cada **observación** tiene su propia **fila**



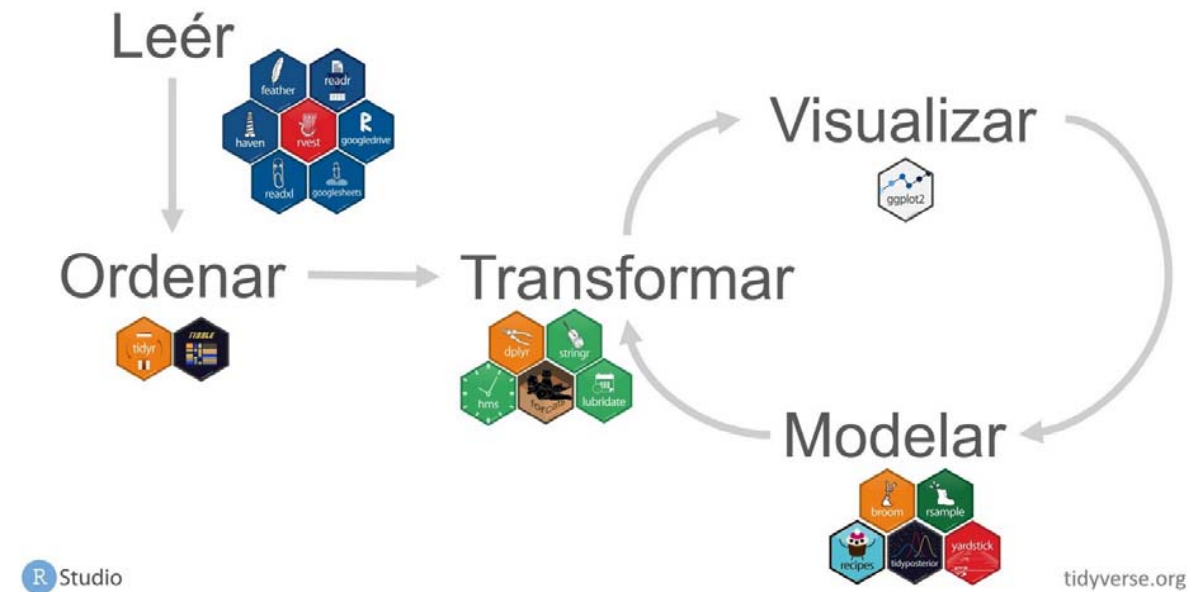
**conductos**  
(pipes)

**x %>% f(y)** se convierte en **f(x, y)**

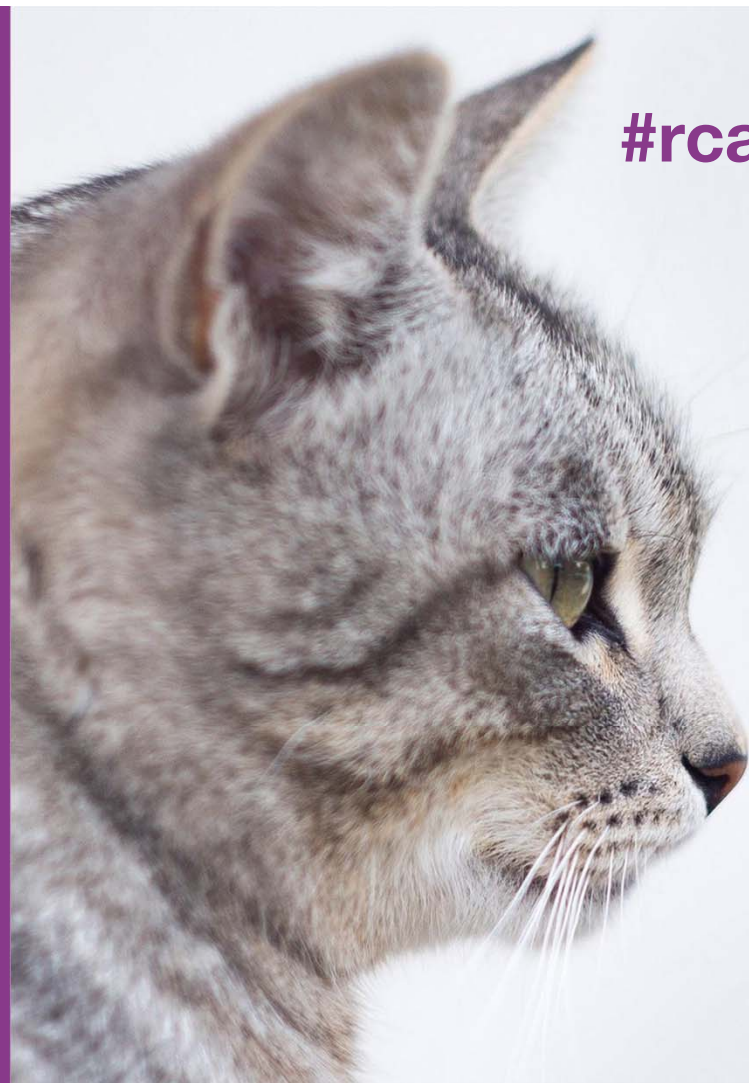
- Nombres de observaciones/filas (en primera columna o por defecto es numero de fila), permite extraer filas por nombre
- Nombres de variables/columnas en primera fila, permite extraer variables por nombre
- No hay observaciones nulas
- Variables nominales u ordinales como factores para agrupar analisis o gráficos



## Paquetes del “tidyverse”



<http://r4ds.had.co.nz/index.html>



## #rcatladies

Como seguir: tutoriales y ayuda

<http://www.sthda.com/english/>

DataCamp: <https://www.datacamp.com/>

Coursera: <https://www.coursera.org/learn/r-programming>

<https://swcarpentry.github.io>

Machetes (Cheat Sheets):

- Data Visualization

- Package Development

- Data Wrangling

- R Markdown

- R Markdown Reference Guide

- Shiny

- R Reference Card

- Writing R extensions

- Google's R Style Guide

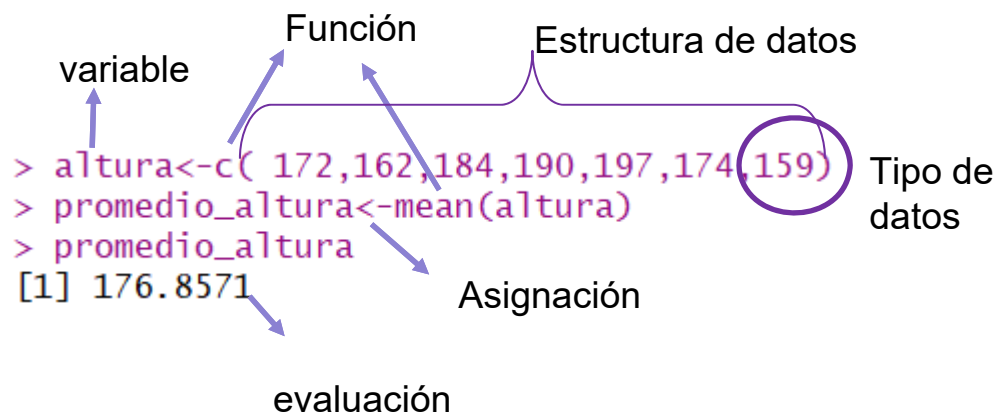




# Elementos de la sintaxis de R

Las entidades que R crea y manipula se denominan objetos. Estos pueden ser de muchos tipos

- Asignación y Evaluación
- Tipos de datos
- Estructuras de datos
- Operadores
- Funciones
- Variables globales vs locales o temporales



# Operadores



Tipos de funciones especiales

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

## Aritméticos

+ adición  
-subtracción  
\* multiplicación  
/ división  
~ potencia  
%% modulo  
%/ % división de enteros

## Comparativos o relacionales

< menor que  
> mayor que  
<= menor o igual que  
>= mayor o igual que  
== igual  
! Diferente

## Lógicos

! X NO lógico  
x & y Y lógico  
x && y id.  
x | y O lógico  
x || y id.  
xor(x,y) O exclusivo







# Funciones

Una función es un conjunto de instrucciones para realizar una tarea específica

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

■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(object)  
}
```

```
miFuncion <- function(x){  
  y <- x + 1  
  return(y)  
}  
myFunction( x = 32 )
```

```
## [1] 33
```



## Exportar/importar un archivo (tabla por ej.)

```
write.csv(x = iris, file = "iris.csv")
```

```
myTable <- read.csv(file = "iris.csv")
```

```
head(myTable)
```

```
myTable2 <- read.csv(file = "http://www.ats.ucla.edu/stat/data/hsb2.csv")
```

