Introducción a **R**

Santiago Caño Muñiz

El objetivo

Aprender a programar es aprender a pensar

En tres días de taller es imposible dominar una herramienta tan versátil como R. Por ello, cuando empecé a preparar este curso me puse un solo objetivo en mente: que sintáis la **potencia** de R. Si consigo eso, entonces, sé que la semilla de la curiosidad os empujará a continuar aprendiendo.

Quién es Santiago

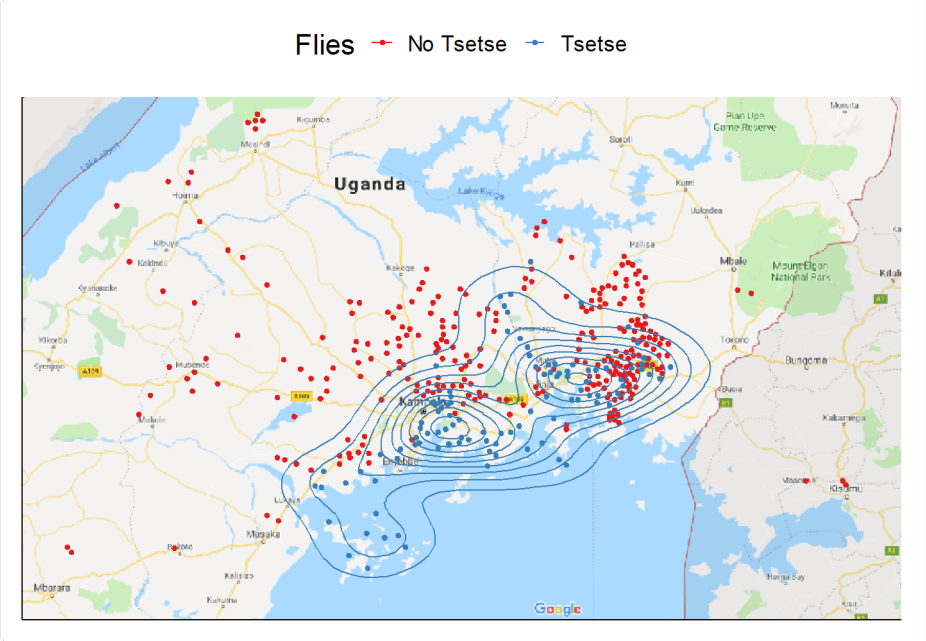
En pocas palabras

- Licenciado en Biología por la **Universidad Autónoma** de Madrid
- **Master** en Biología Molecular y Biotecnología por la Universidad de Groningen
  - Descubrí R a principios de mi primer proyecto de investigación.
  - Representación de datos, regresión lineal y no lineal, análisis de balance de flujos (FBA, por sus siglas en inglés)
- Estudiante de doctorado en el Laboratorio de Biología Molecular, **Cambridge**
  - Regresión con Extensión de modelos lineales (GLM, por sus siglas en inglés)
  - Machine Learning para clasificación de señales
- Escalador de cimas y bachatero en mi tiempo libre

¿Por qué aprender R?

Una forma de representar ideas

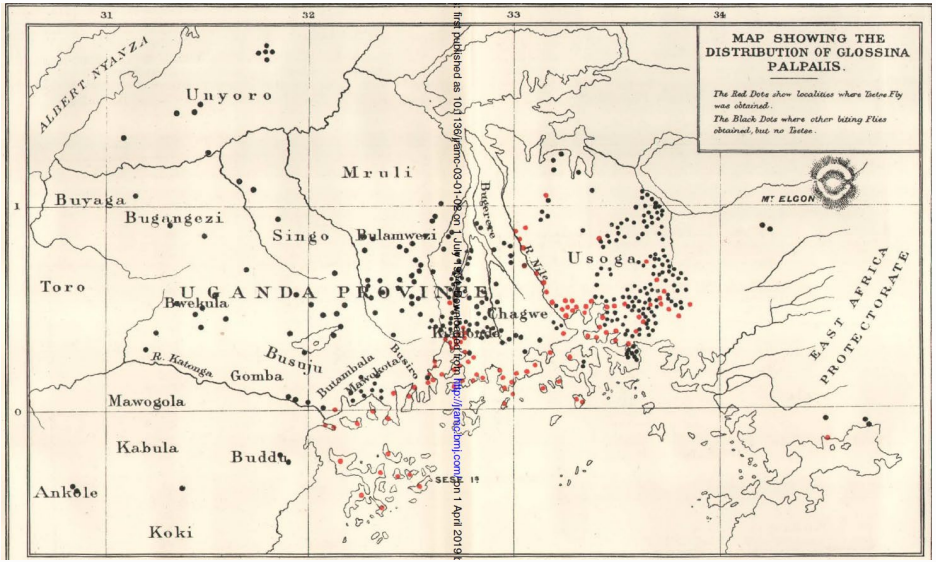
Age	Sex	District	Shamba	Trypanosoma	Filaria
25	M	Sese Island	Sewana	+	+
20	M	Sese Island	Kaganda I	-	+
25	M	Sese Island	Semagala I	+	+
30	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
25	M	Sese Island	Buvovu I	-	+
25	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
30	M	Sese Island	Buvu I	-	+
35	M	Sese Island	Semagala I	-	-
20	M	Sese Island	Semagala I	-	-
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
25	M	Sese Island	Semagala I	+	-
35	M	Sese Island	Bunami I	+	+



¿Por qué aprender R?

Una forma de representar ideas

Age	Sex	District	Shamba	Trypanosoma	Filaria
25	M	Sese Island	Sewana	+	+
20	M	Sese Island	Kaganda I	-	+
25	M	Sese Island	Semagala I	+	+
30	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
25	M	Sese Island	Buvovu I	-	+
25	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
30	M	Sese Island	Buvu I	-	+
35	M	Sese Island	Semagala I	-	-
20	M	Sese Island	Semagala I	-	-
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
25	M	Sese Island	Semagala I	+	-
35	M	Sese Island	Bunami I	+	+



Dr D. Bruce, 1903

¿Por qué aprender R?

Una forma de representar ideas

Age	Sex	District	Shamba	Trypanosoma	Filaria
25	M	Sese Island	Sewana	+	+
20	M	Sese Island	Kaganda I	-	+
25	M	Sese Island	Semagala I	+	+
30	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
25	M	Sese Island	Buvovu I	-	+
25	M	Sese Island	Kaganda I	-	+
20	M	Sese Island	Semagala I	-	+
30	M	Sese Island	Buvu I	-	+
35	M	Sese Island	Semagala I	-	-
20	M	Sese Island	Semagala I	-	-
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
25	M	Sese Island	Semagala I	+	-
35	M	Sese Island	Bunami I	+	+

```r
library(ggplot2)
library(magick)
library(readxl)
library(ggpubr)
library(data.table)

map_u <- image_read("Figures/Mapa_uganda.png")

Sick_coord <- read_xlsx(path = "Datasets/D_Bruce_map_dataset_v1.xlsx",
                        sheet = "Sickness") %>% data.table
Flies_coord <- read_xlsx(path = "Datasets/D_Bruce_map_dataset_v1.xlsx",
                         sheet = "Flies") %>% data.table

# Plot 1

image_ggplot(map_u) +
  geom_point(data = Sick_coord, mapping = aes(x = X , y = Y, col = Sickness)) +
  geom_density_2d(data = Sick_coord[Sickness == "Present"],
                  mapping = aes(x = X , y = Y, col = Sickness)) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "", y = "") +
  theme_pubr(legend = "top", base_size = 20) +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
# Plot 2

image_ggplot(map_u) +
  geom_point(data = Flies_coord, mapping = aes(x = X , y = Y, col = Flies)) +
  geom_density_2d(data = Flies_coord[Flies == "Tsetse"],
                  mapping = aes(x = X , y = Y, col = Flies)) +
  scale_color_brewer(palette = "Set1") +

  labs(x = "", y = "") +
  theme_pubr(legend = "top", base_size = 20) +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Aprender a programar es aprender a pensar

R es un lenguaje que nos ayuda a representar nuestra visión del mundo

Que es R

Una herramienta para hacernos la vida más fácil

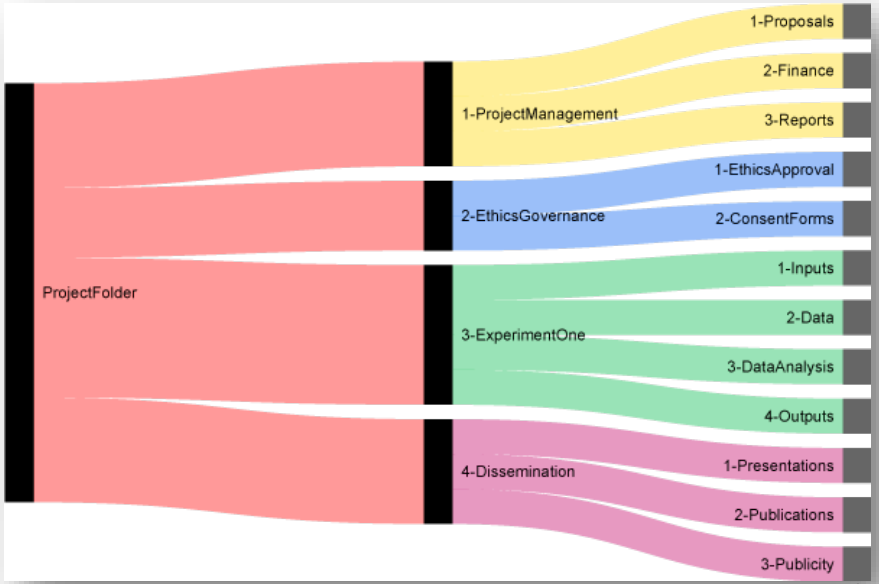
- **Entorno de análisis de datos dinámico**
- Open source
- Comunidad muy activa
- Infinidad de paquetes con funciones

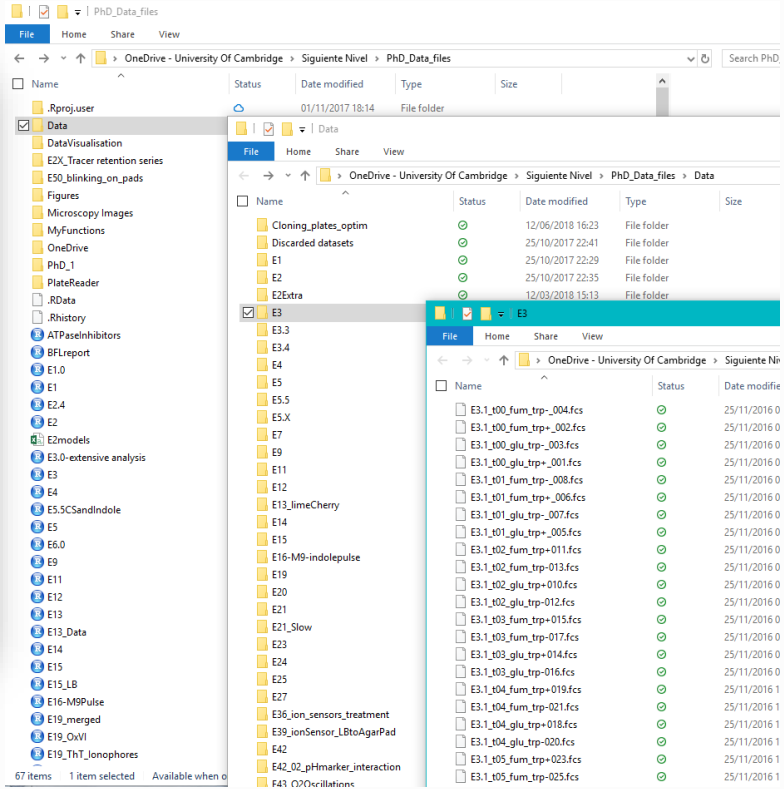
Que es R

Mi primera experiencia con R

Organizar los archivos

Una jerarquía de archivos



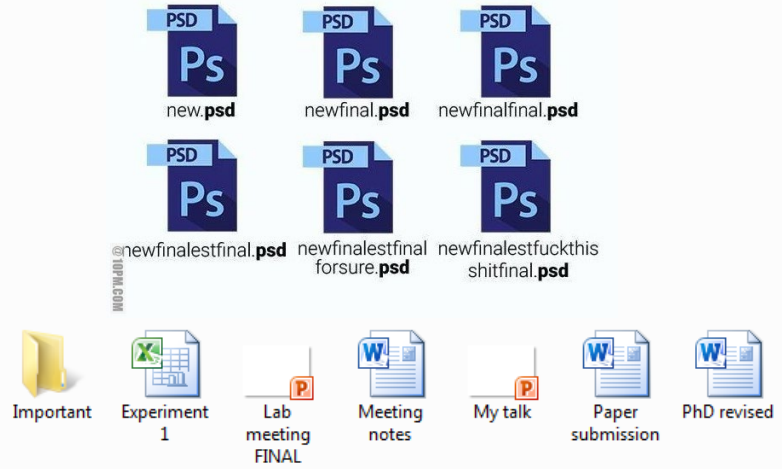


® http://nikola.me/folder_structure.html

Organizar los archivos

Nombrar archivos… ¿¿de verdad importa??

Copyright: http://10pm.com/



¿Recordaremos que significa dentro de… 3 años?

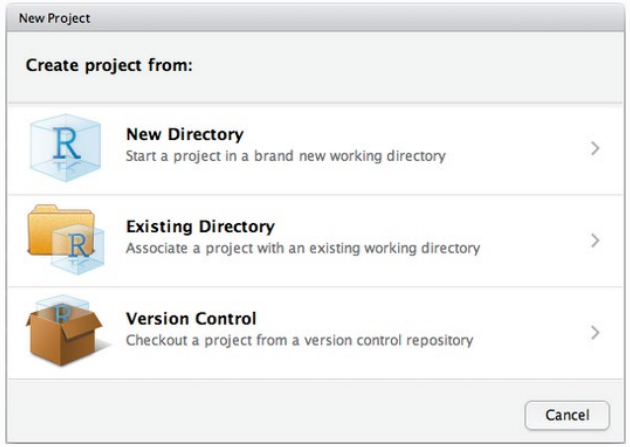
Lo que yo hago

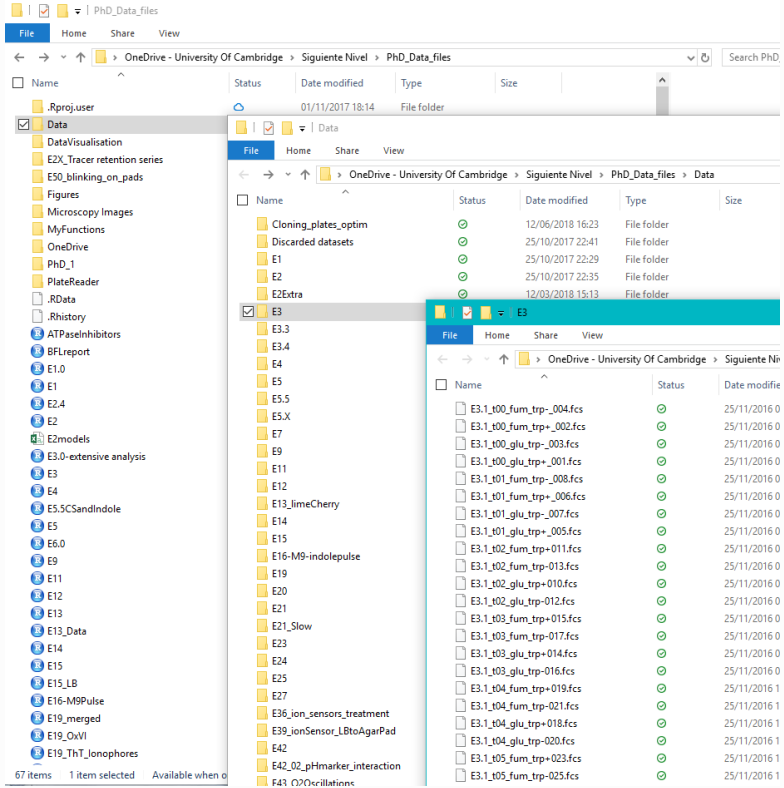
EXP_AAAAMMDD_NOMBRE_VX.X.csv

Evita usar simbolos ?,$,%,^,&,*,(,),-,#,?,,,<,>,/,|,\,[,],{, and };

RStudio

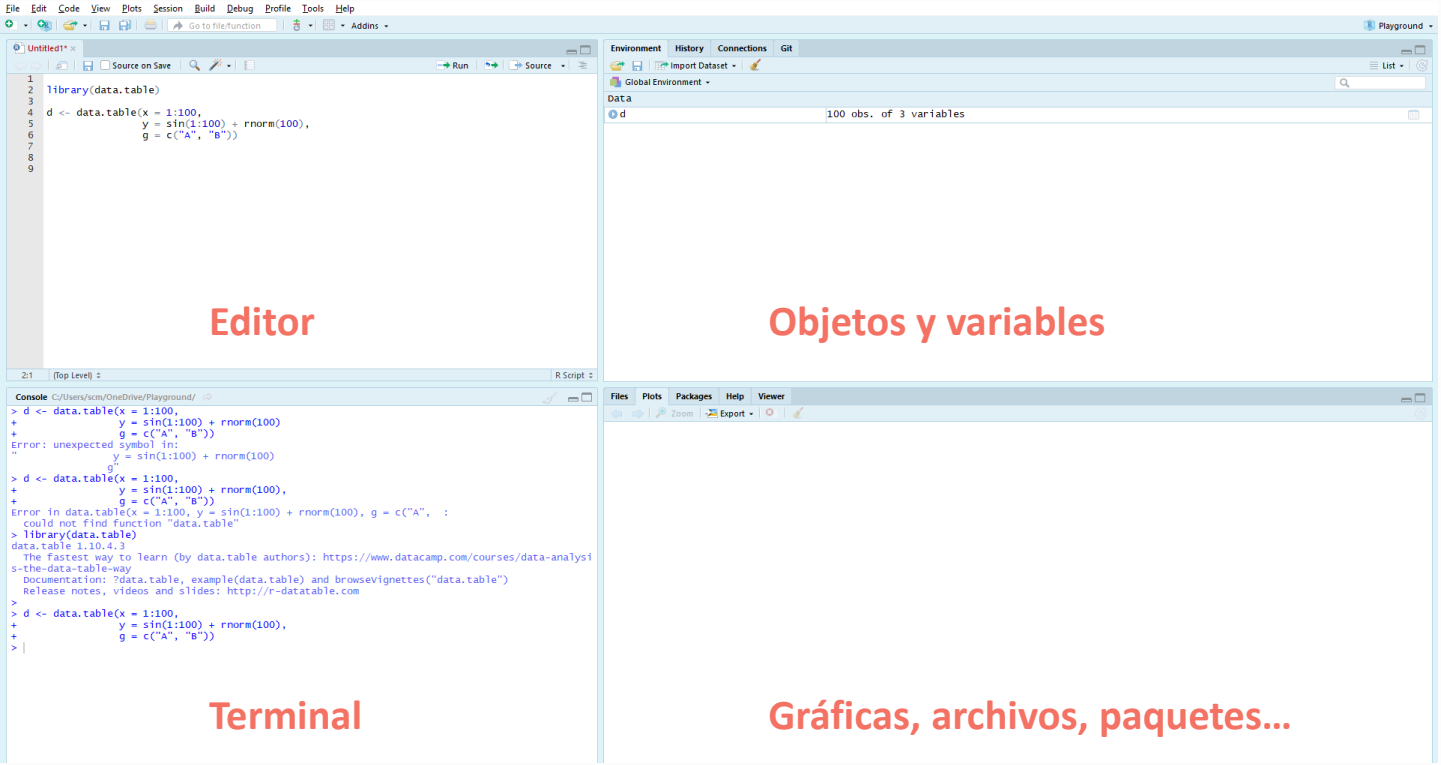
Como reflejar esto en la organización de la información





Dentro de R

RStudio



La base de R

Herencia del lenguage matemático

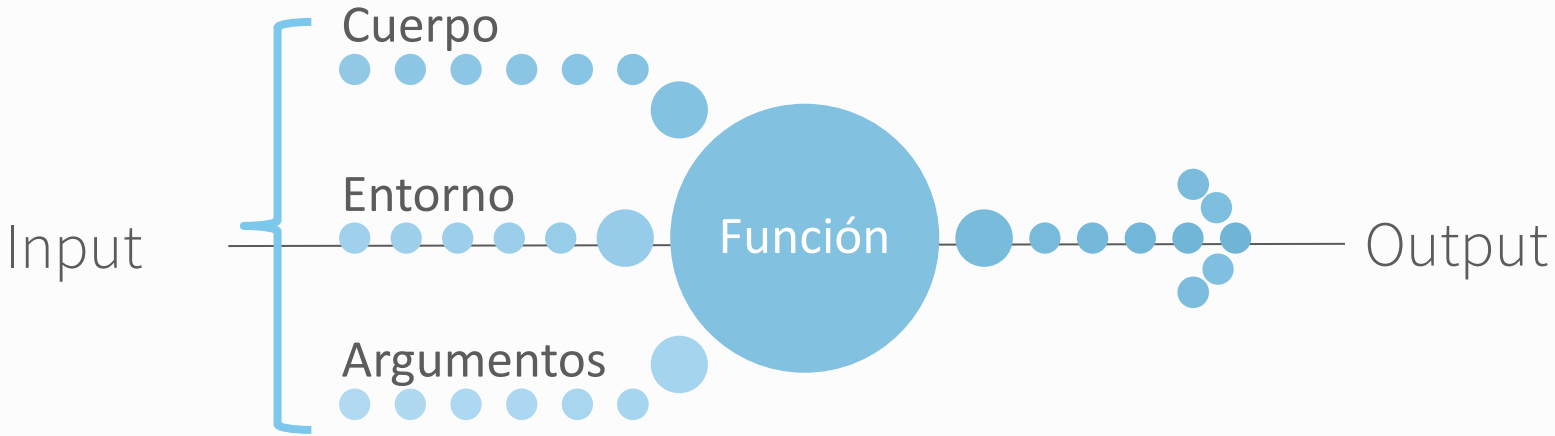
Input $f(x)$ Output

$$A = f(r) = \pi r^2$$

$$S = \frac{dA}{dr} = \frac{df(r)}{dr} = 2\pi r$$

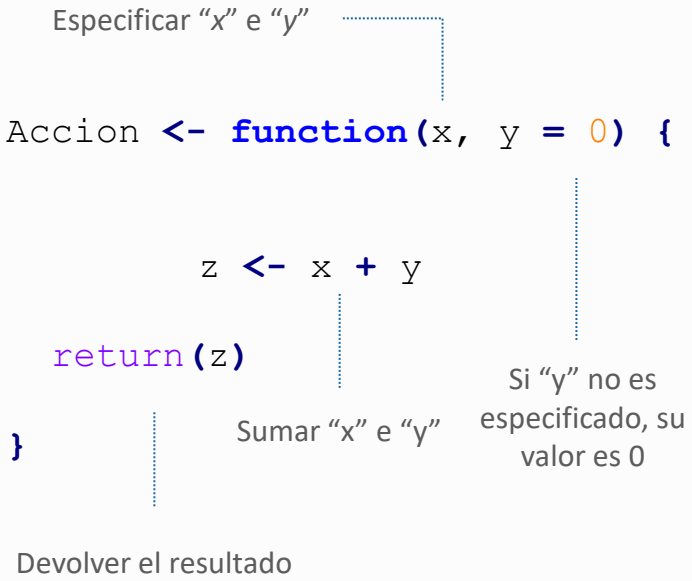
La base de R

El lenguaje funcional



R es una super-calculadora con esteroides

Pero… que es una función?



Los 3 elementos

- **Argumentos**: lista de elementos, especificados por **orden** o **nombre.**
- **Cuerpo:** Operaciones entre corchetes que **transforman** los argumentos.
- **Entorno**: Variables accesibles al cuerpo de la función.

R es una super-calculadora con esteroides

Pero… que es una función?

```
# Introducción a funciones

runif(n = 10, min = 3, max = 10)     # Generar números aleatorios, todos los argumentos declarados
runif(10, 3, 10)                     # Declarar argumentos usando la posición

runif(n = 10)                        # Declarar n, usar min & max por defecto
runif(10)                            # Declarando n por posición y min & max por defecto

runif(min = -1, n = 3)               # Orden alterado

# Aritmética básica
1 + 8

2 - 9

2 * 3

5/3
```

R es una super-calculadora con esteroides

Operaciones lógicas

```
x == y                  # Igual: x igual a y
identical(x, y)         # Versión funcional de ==
!x                      # Negación de x
x & y                   # Conjunción "Y": x e y
x | y                   # Conjunción O: x o y
x < y                   # Menor que: X menor que y
x > y                   # Mayor que: X mayor que y
x <= y                  # Menor o igual que: x menor o igual que y
x >= y                  # Mayor o igual que: x mayor o igual que y
x != y                  # Different: x es diferente a y
xor()                   # Conjunción O
isTrue()                # Conjunción O
```

R es una super-calculadora con esteroides

En R existe una función para todo

```
A <- ...                        # Función para asignar una variable
c(object,object,...)            # Concatenar variables en un vector
length(object)                  # Número de elementos en un objeto
str(object)                     # Mostrar estructura de un objeto
class(object)                   # Extraer clase o tipo de objeto
names(object)                   # Nombres
rm(object)                      # Eliminar un objeto
mean(vector)                    # Media
median(vector)                  # Mediana
sd(vector)                      # Desviación estándar
sqrt(vector)                    # Raíz cuadrada
log(vector)                     # Logaritmo
exp(vector)                     # Exponente

summary()                       # Resumen
getwd()                         # Directorio de tabajo
read.delim()                    # Importar tabla, también read.csv, read.csv2, read.txt, read.table
```

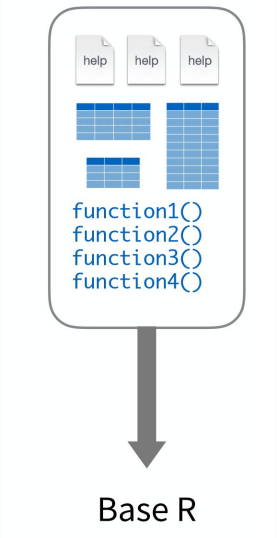
A programar

Por ejemplo

```
mean(1:5, 0.1)
mean(x = 1:5, trim = 0.1)
mean(1:5, trim = 0.1)
mean(x = 1:5, 0.1)
```

Los paquetes en R

Las funciones de R vienen en paquetes



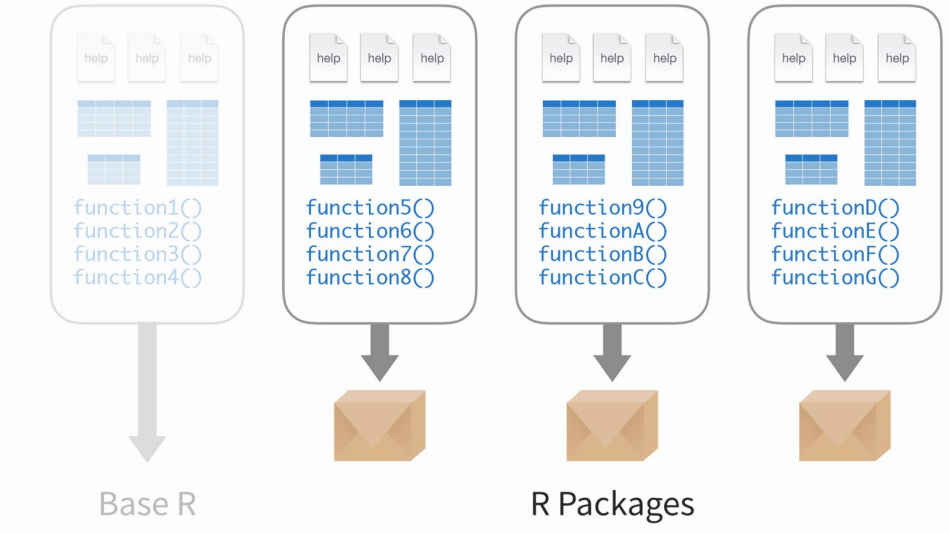
Que son

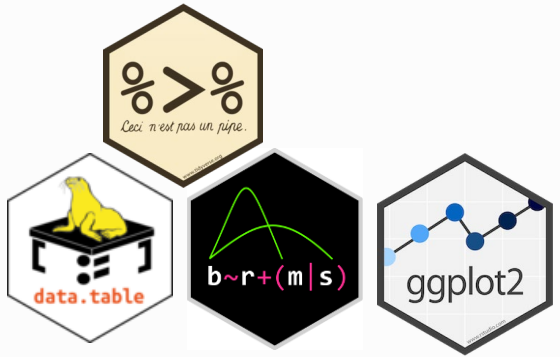
- La comunidad de R que desarrolla funciones relacionadas las agrega en una librería que otros usuarios puede descargar e instalar.
- Esto nos permite empezar nuestro trabajo donde otros terminaron y concentrar nuestro esfuerzo en una solo problema.
- Suelen venir acompañados de libros de instrucciones llamados “**vignette**” que nos explican como usar las funciones de la librería.

```
install.package(data.table)                          # Instalar libreria desde CRAN
devtools::install_github("/paul-buerkner/brms") # Instalar desde GitHub


library(tidyverse)                                   # Cargar libraria
data.table::melt.data.table()          # Llamar una función sin cargar la libreria
```

Los paquetes en R

Las funciones de R vienen en paquetes





- En este taller trabajaremos con los siguientes librerías: data.table, magrittr, ggplot, glmmTMB, mgcv
- Cuidado con usar demasiadas librerías
  - Solapamiento de funciones con el mismo nombre
  - Reproducibilidad en otras computadoras

Primer contacto

Representar objetos matemáticos

Vectores

$$X = [x_1, x_2, x_3, x_i \dots \dots x_n]$$

Numeric: $\mathbb{R}$

- 0, -8, 3.14…

Logic: 1|0

- TRUE, FALSE, T, F

Character

- “Que”, “es”, “eso”, “Eso es Queso”

```
1 + 1                    # Almoadilla para anotar tu código
Nombre_x <- valor        # Definir objetos con el operador <-
y <- 1
```

Principios para nombrar una variable

- Empezar por una letra, sin carateres irregulars `(&, %, ^, …)`
- Breve y descriptivo. Para nombres compuestos conectar con _

```
`<-`(x, 2)               # Asignar
x <- y <- 1              # Asignar multiples elementos
y <- c(1, -2, 8, 5, 5e5)# Definir manualmente un vector
z <- c("A", "B", "C")    # Vectores de carteres
x <- c(FALSE, TRUE, F, T) # Vectores, carteres
x <- c(1L, 2L, 43L)
```

Primer contacto

Representar objetos matemáticos

Vectores

```
x + y
x * 3
x / y
x^3
1:5                                  # Dos puntos ":" para indicar secuencias
5:1
y <- c(1, -2, 8, 5, 5e5)
y[3]                                 # Extraer elementos
[1] 8
y[-2]                                # Numeros negativos para excluir elementos
[1] 1  8  5  5e5
y[2:3]                               # Extraer secuencia de elementos
# Nombrar elementos para crear diccionarios
y <- c("a" = 1, "b" = -2, "c" = 8, "d" = 5, "e" = 5e5)
y["d"]
[1] 5
y[y > 5]                             # Extraer elementos usando test lógicos ( también ==, >=, <= )
   c     e
8e+00 5e+05
```

Primer contacto

Rrepresentar objetos matemáticos

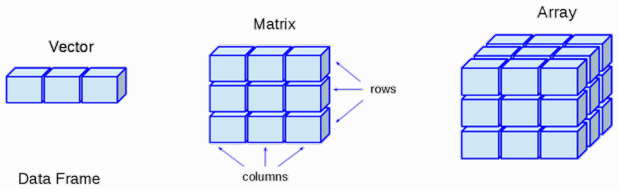
Reciclado

```
# Si dos vectores tienen distinta longitud, el más corto es reciclado reciclará hasta igualar la
longitud del más largo con el vector más largo.

x = c(10, 20, 30)
y = c(1, 2, 3, 4, 5, 6, 7, 8, 9)

> y + x
# R usa "y" como vector de 9 elementos y "x" lo repetirá 3 veces

[1] 11 22 33 14 25 36 17 28 39
```

Primer contacto

Rrepresentar objetos matemáticos

Estructuras homogeneas



Principios para nombrar una variable

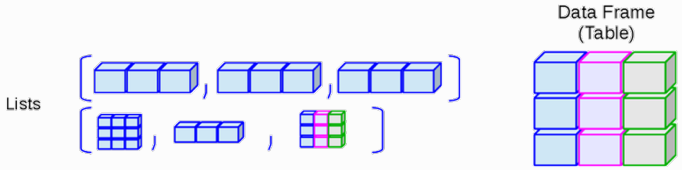
- Dentro de R, una matriz es un vector con dos atributos extra:
  - Filas
  - Columnas

```
mat <- matrix(c(1, -2, 8, 5, 7, 0, 3, 6, 9),
              nrow = 3, ncol = 3)

mat[4]                    # Usar matriz como vector
[1] 5

mat[1, 2]                 # mat[fila, columna]
[1] 5

mat[ ,c(1, 3)]
     [,1] [,2]
[1,]    1    3
[2,]   -2    6
[3,]    8    9

A <- array(data = 1:27, dim = c(3, 3, 3))
A[ 2, 2, 3]
[1] 23
```

Primer contacto

Rrepresentar objetos matemáticos

Estructuras heterogeneas




```
mat <- matrix(c(1, -2, 8, 5), nrow = 2, ncol = 2)
mat[1, 2]                              # mat[fila, columna]
[1] 8
lista <- list("a" = c(1, 4, 5, 2),
              "b" = c("vaca", "cerdo", "gallo"))

lista[["a"]][1]           # [[ para acceder a cada elemento
lista$a[1]                # $ para acceder a cada element
```

Listas y tablas

- Contiene estructuras de diferentes tipos , o incluso contener una lista
- Pueden ser también de diferente longitud

A programar

Por ejemplo

```
x + y
x * 3
x / y
x^3
1:5                                  # Dos puntos ":" para indicar secuencias
5:1
y <- c(1, -2, 8, 5, 5e5)
y[3]                                 # Extraer elementos
[1] 8
y[-2]                                # Numeros negativos para excluir elementos
[1] 1  8  5  5e5
y[2:3]                               # Extraer secuencia de elementos
# Nombrar elementos para crear diccionarios
y <- c("a" = 1, "b" = -2, "c" = 8, "d" = 5, "e" = 5e5)
y["d"]
[1] 5
y[y > 5]                             # Extraer elementos usando test lógicos ( también ==, >=, <= )
   c     e
8e+00 5e+05
```

Tablas

Data.fame

- A nivel intuitivo, `data.frame` es la forma natural de representar información en nuestra mente.
- Cada columna representa un vector
- Cada fila un caso

Field Name	Area	Slope	Vegetation
Nash's Field	3.6	11	Grassland
Silwood Bottom	5.1	2	Arable
Nursery Field	2.8	3	Grassland
Rush Meadow	2.4	5	Meadow
Gunness' Thicket	3.8	0	Scrub
Oak Mead	3.1	2	Grassland
Church Field	3.5	3	Grassland

```
tabla <- data.frame(Field_N = c("Nash`s", "Silwood", "Nursery",
                    "Rush", "Gunness", "Oak Mead", "Church Field"),
                       Area = c(3.6, 5.1, 2.8, 2.4, 3.8, 3.1, 3.5),
                       Slope = c(11, 2, 3, 5, 0, 2 , 3),
                       Vegetation = c("Grass", "Arabl", "Grass",
                      "Meadow", "Scrub", "Grass", "Grass"))

# Crear agregando vectores pre-existentes

tabla <- data.frame(Field_N, Area, Slope, Vegetation)

# Unir tablas
tabla <- cbind(Field_N, Area, Slope, Vegetation)
Tabla <- rbind(tabla_1, tabla_2)

# También vía read.*
tabla <- read.csv("ruta/a/mi/archivo")
tabla <- read.txt("ruta/a/mi/archivo") # read.*
```

Tablas

Data.fame

- A nivel intuitivo, `data.frame` es la forma natural de representar información en nuestra mente.
- Cada columna representa un vector
- Cada fila un caso

Field Name	Area	Slope	Vegetation	Soil pH
Nash's Field	3.6	11	Grassland	4.1
Silwood Bottom	5.1	2	Arable	5.2
Nursery Field	2.8	3	Grassland	4.3
Rush Meadow	2.4	5	Meadow	4.9
Gunness' Thicket	3.8	0	Scrub	4.2
Oak Mead	3.1	2	Grassland	3.9
Church Field	3.5	3	Grassland	4.2

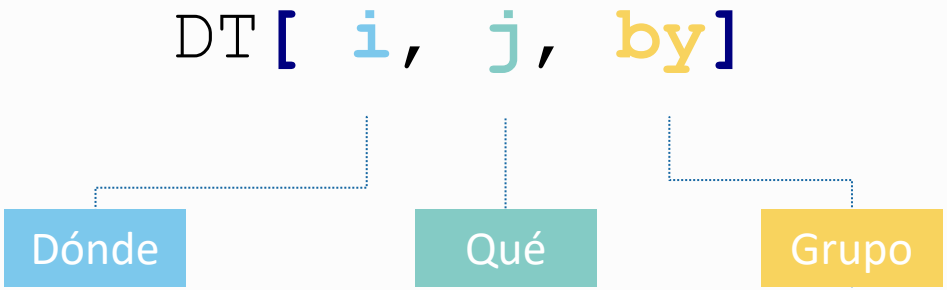
```
head(tabla)                 # Explorar primeros/últimos valores
tail(tabla)
str(tabla)

tabla$Area                  # Exponer los vectores con tabla$
mean(tabla$Slope)

summary(tabla)

tabla[1, 2]                 # Acceder a los valores como una matriz
tabla[, 2:3]
tabla[, "nombre"]           # Utilizar el nombre de la columna

tabla$coste                 # Exponer los vectores
```

Data.table

Una expansión a los data.frame

Propiedades

- La sintaxis y el uso es similar a los `data.frame`
- Es extremadamente rápido
- Ofrece herramientas para:
  - Agregado de datos
  - Actualizar celdas
  - Unir tablas
- Permite una notación elegante
- **Sin dependencias**



Data.table

Continuación

```
matrix(data.table)
d <- data.table(mtcars, keep.rownames = TRUE)
```

```
 >                   rn  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
 1:           Mazda RX4 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
 2:       Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
 3:          Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
 4:      Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
 5:   Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
 6:             Valiant 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
 7:          Duster 360 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
```

```
# ¿Cómo realizar operaciones de extracción sencillas?

d[cyl > 4]                                      # Filtrar con un test lógico simple

d[rn %in% c("Mazda RX4", "Hornet Sportabout")]  # Elementros dentro de un vector

d[rn %like% "Mazda"]                            # Extraer elementos similares
```

Data.table

Continuación

```
# Agregar datos

d[, .(Sumario = mean(cyl)),          # Por grupos
            by = gear]

d[, .(Sumario = mean(cyl)),
            by = .(gear, vs)]        # Multiples grupos

d[, Media_hp := mean(hp),            # Multiples-grupos y crear una columna nueva
            by = .(gear, vs)]

d[, .("Media_hp", "Media_cyl"):=     # Multiples-grupos y múltiples columnas
            .(mean(hp),
              mean(cyl),
            by = .(gear, vs)]

d[, mean(wt[vs == 0])/mean(wt[vs == 1])] # Vector dentro de una columna
```

A programar

Por ejemplo

```
# Agregar datos
d <- data.table(mtcars)
d[, .(Sumario = mean(cyl)),          # Por grupos
            by = gear]

d[, .(Sumario = mean(cyl)),
            by = .(gear, vs)]        # Multiples grupos

d[, Media_hp := mean(hp),            # Multiples-grupos y crear una columna nueva
            by = .(gear, vs)]

d[, .("Media_hp", "Media_cyl"):=     # Multiples-grupos y múltiples columnas
            .(mean(hp),
              mean(cyl),
            by = .(gear, vs)]

d[, mean(wt[vs == 0])/mean(wt[vs == 1])] # Vector dentro de una columna
```

Los conectores

Avanzando de A %>% B

El operador %>%

- Uno de los elementos más útiles y poderosos de R.
- El operador ¨%>% ayuda a estructurar el código y minimiza la creación de "variables transitorias.
- Requiere el paquete `magrittr` o `tidyverse`.
- Las ideas básicas son:
  - `x %>% f` equivale a `f(x)`
  - `x %>% f %>% g %>% h` equivale a `h(g(f(x)))`

Marcador

```
Objeto %>%
    función1(.) %>%
    función2(.) ->
    resultado
```

Los conectores

Avanzando de A %>% B

El operador %>%

- Uno de los elementos más útiles y poderosos de R.
- El operador ¨%>% ayuda a estructurar el código y minimiza la creación de “variables transitorias.
- Requiere el paquete `magrittr` o `tidyverse`.
- Las ideas básicas son:
  - `x %>% f` equivale a `f(x)`
  - `x %>% f %>% g %>% h` equivale a `h(g(f(x)))`

Ejemplo: Representar medias por grupo

Pila de variables “muertas”

```
# Extraer valores
Subset <- tabla[ tabla$a > "valor_x"]
# Sumarizar la información
mus <- aggregate(test ~ condicion, Subset, FUN = mean)
stds <- aggregate(test ~ condicion, Subset, FUN = sd)
resumen_Ss <- merge(mus, stds, by = "condicion")
```

```
 # Crear la gráfica
colnames(resumen_Ss)<- c("condicion","mus","stds")
ggplot(resumen_Ss, aes(x = condicion, y = mus)) +
       geom_point() +
       geom_errorbar(aes(ymin = mus - stds,
                         ymax = mus +    stds)) +
       theme_bw()
```

Los conectores

Avanzando de A %>% B

El operador %>%

- Uno de los elementos más útiles y poderosos de R.
- El operador ¨%>% ayuda a estructurar el código y minimiza la creación de “variables transitorias.
- Requiere el paquete `magrittr` o `tidyverse`.
- Las ideas básicas son:
  - `x %>% f` equivale a `f(x)`
  - `x %>% f %>% g %>% h` equivale a `h(g(f(x)))`

Ejemplo: Representar medias por grupo **Código Imposible de leer**

```
resumen_Ss <- merge(aggregate(test ~ condicion,
                              tabla[, tabla$a > "valor_x"],
                        FUN = mean),
                  aggregate(test ~ condicion,
                            tabla[, tabla$a > "valor_x"],
                        FUN = sd), by = "condicion")
 # Crear la gráfica
colnames(resumen_Ss)<- c("condicion","mus","stds")
ggplot(resumen_Ss, aes(x = condicion, y = mus)) +
       geom_point() +
       geom_errorbar(aes(ymin = mus - stds,
                         ymax = mus +    stds)) +
       theme_bw()
```

Los conectores

Avanzando de A %>% B

El operador %>%

- Uno de los elementos más útiles y poderosos de R.
- El operador ¨%>% ayuda a estructurar el código y minimiza la creación de "variables transitorias.
- Requiere el paquete `magrittr` o `tidyverse`.
- Las ideas básicas son:
  - `x %>% f` equivale a `f(x)`
  - `x %>% f %>% g %>% h` equivale a `h(g(f(x)))`

Ejemplo: Representar medias por grupo

```
tabla[a > "valor_x" ] %>% # Filtro
    .[, .(mus = mean(test),
          stds = sd(test),
       by = condicion] %>% # Calcular parámetros

# Representación gráfica
ggplot(., aes(x = condicion, y = mus) +
       geom_point()
       geom_errorbar(aes(ymin = mus - stds,
                         ymax = mus + stds) +
       theme_bw()
```

A programar

Por ejemplo

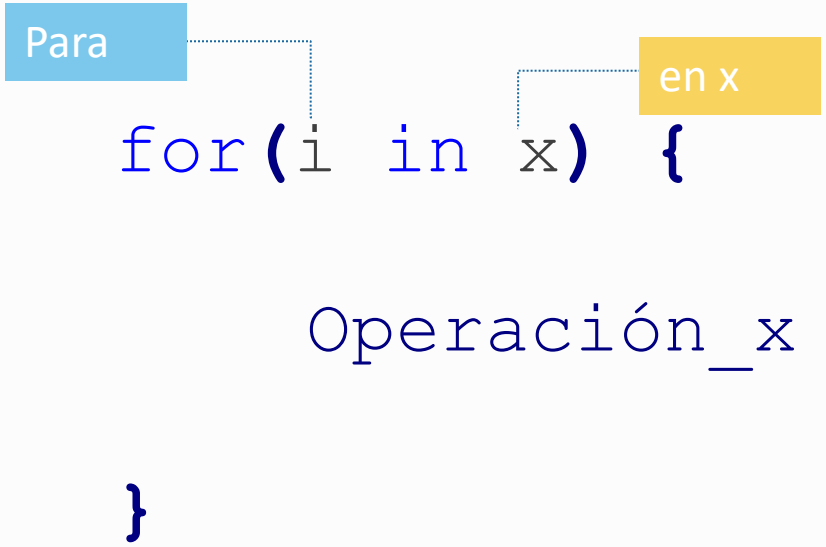
```
library(tidyverse)
library(data.table)
data("mtcars") # Load data
d <- data.table(mtcars, keep.rownames = T)

d[, cyl > 3] %>%
            ggplot(., aes(x = disp, y = wt)) +
            geom_point() +
            theme_bw

d %>%
  lm(formula = disp ~ wt, data = .) %>%
  summary
```

Iteraciones y funcionales

La esencia de la programación

¿Que son?

- Las computadoras son especialmente útiles cuando la tarea requiere de repetición
- R nos provee de tres herramientas básicas para repetir accione:
  - `For`
  - `Repeat`
  - `La familia *apply`
- Además, es posible filtar con test lógicos:
  - `If`
  - `ifelse`



Iteraciones y funcionales

Ejemplo

```
# Generate a sample dataset

set.seed(2018)

d <- data.frame(replicate(6, sample(c(1:10, -99), 100, rep = TRUE)))
names(d) <- letters[1:6]
head(d)
 a   b   c  d   e  f
1 4   7 -99  9   9  2
2 6   2   8 10   6 10
3 1 -99   9  3   4  1
4 3   7   7  7 -99  6

# ¿Cómo calculo la media de cada columna?
mean(d$a)
mean(d$b)
mean(d$b)
mean(d$d)
mean(d$e)
```

Iteraciones y funcionales

Ejemplo

```
# Con iteración-for

for (i in 1:ncol(d)) {              # Usando ":" para generar secuencia de 1 al número de columnas

            x <- mean(d[, i])
            print(x)
            }

# Si queremos guardar el resultado, primero creamos un vector (u otro formato) vacío

medias <- rep(NA, ncol(d))

for (i in seq_along(medias)) {

            x <- mean(d[, i])       # Iteramos por columna
            medias[i] <- x          # Guardamos el resultado en el vector "medias", posición "i"

}
```

Iteraciones y funcionales

La esencia de la programación

¿Que son?

- Las computadoras son especialmente útiles cuando la tarea requiere de repetición
- R nos provee de tres herramientas básicas para repetir accione:
  - `For`
  - `La familia *apply`
- Además, es posible filtar mediante test lógicos:
  - `If`
  - `ifelse`

```
*apply(x, Fun = f(x) )
```

Aplicar al objeto x, la función f

Iteraciones y funcionales

Ejemplo

```
# Con iteración-*apply

medias <- apply(X = d, MARGIN = 2, FUN = mean)

> medias
    a     b     c     d     e    f
-4.26 -4.75 -1.85 -5.92 -4.43 1.40

# Si además queremos especificar otros argumentos, podemos indicarlos al final

apply(X = d, MARGIN = 2, FUN = quantile, probs = c(0.1, 0.5, 0.9))

>    a  b c   d  e  f
10% 1 -9 1 -99 -9  1
50% 4  5 5   5  6  5
90% 9  9 9   9 10 10
```

Iteraciones y funcionales

La esencia de la programación

¿Que son?

- Las computadoras son especialmente útiles cuando la tarea requiere de repetición
- R nos provee de tres herramientas básicas para repetir accione:
  - `For`
  - `La familia *apply`
- Además, es posible filtar mediante test lógicos:
  - `If`
  - `ifelse`

```
ifelse(test = ***,
       yes = Accion_A,
       no = Accion_B)
```

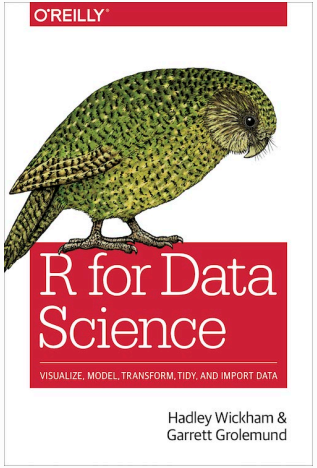
Iteraciones y funcionales

Ejemplo

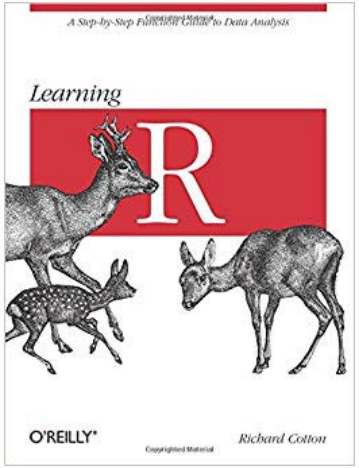
```
# Con iteración-*apply
d <- apply(d, 2, function(x) {ifelse(x == -99, NA, x)})
medias <- apply(X = d, MARGIN = 2, FUN = mean)

# Con iteración-*apply

medias <- apply(d, 2, function(x) {tmp <- ifelse(x == -99, NA, x)}) %>%
            apply(X = ., MARGIN = 2, FUN = mean, na.rm = TRUE)
medias

# Con iteración-*apply

medias <- apply(d, 2, function(x) {tmp <- ifelse(x == -99, NA, x)
                                   mean(tmp, na.rm = TRUE)})
> medias
      X1       X2       X3       X4       X5       X6
5.585106 5.369565 5.263736 5.866667 5.423913 5.217391
```

Canales de apoyo

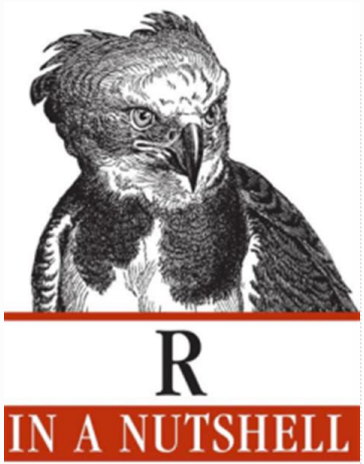
Recursos educativos del sXXI



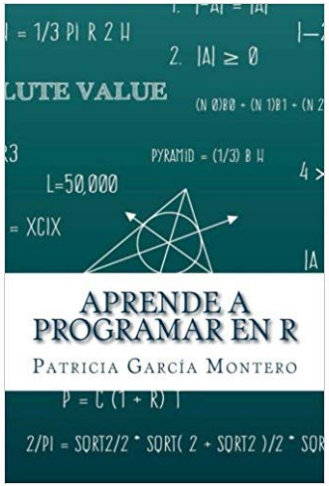
R for Data Science,
H. Wickham &
G. Grolemund



Learning R,
R. Cotton



R in a nutshell,
J. Adler



Aprender a
programar en R
P. García Montero

¡Gracias por vuestro tiempo!

¿Preguntas?