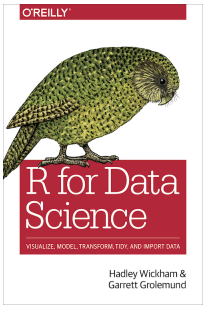
Introducción a R y Rstudio

Introduccion a R y Rstudio

- Es un lenguaje de programación y un entorno para análisis estadístico .
- Fue inicialmente escrito por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en Nueva Zelanda.
- R actualmente es el resultado de un esfuerzo de colaboración de personas del todo el mundo.
- Tiene la posibilidad de modificación directa del código fuente.
- Por otra parte, R es un proyecto GNU similar a S, desarrollado éste por los Laboratorios Bell. Las diferencias entre R y S son importantes, pero la mayoría del código escrito para S corre bajo R sin modificaciones

R Studio

Figure 1: https://www.rstudio.com/



Por otro lado, R Studio es un IDE o entorno de desarrollo integrado. Un Programa para manejar R

Rstudio Cloud https://rstudio.cloud/

Porqué usar R?

- Gratis

- Reproducibilidad
- Novedad
- Popularidad
- Comunidad

Cuando abres por primera vez el RStudio, serás recibido por tres paneles:

- La consola interactiva de R (a la izquierda)
- Ambiente/Historial (en la esquina superior derecha)
- Archivos/Gráficos/Paquetes/Ayuda/Visor (abajo a la derecha)
- Si abres archivos, como los scripts R, también se abrirá un panel de editor en la esquina superior izquierda.

Flujo de trabajo dentro de RStudio

- Probar y jugar dentro de la **consola interactiva de R** y luego copiar el código en un archivo .R para ejecutarlo más tarde.
- Comienza a escribir un **archivo en .R** y usa las teclas de acceso directo de RStudio para ejecutar la línea actual, las líneas seleccionadas o modificadas en la consola interactiva.

Esta es una buena forma de comenzar; todo tu código estará guardado para después. Podrás ejecutar el archivo que quieres desde RStudio o mediante la función source () de R.

Usando R como calculadora

```
2+4
```

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

```
sqrt(45)*234+324/7
```

```
## [1] 1616.005
```

```
sin(1)
```

```
## [1] 0.841471
```

Comparando

```
1 == 1  # igualdad (observa dos signos iguales, se lee como "es igual a")
```

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

```
1 != 2  # desigualdad (leída como "no es igual a")
```

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

```
1 > 0  # mayor que
```

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

Variables y asignaciones

```
A<-34
A
```

```
## [1] 34
```

```
B<-25
```

```
x <- 1/40
```

```
x <- x + 1 # observa cómo RStudio actualiza la descripción de x en la pestaña superior derecha
y <- x * 2
y
```

```
## [1] 2.05
```

Variables y asignaciones

- Los nombres de las variables pueden contener letras, números, guiones bajos y puntos.
- No pueden comenzar con un número ni contener espacios en absoluto.
- Diferentes personas usan diferentes convenciones para nombres largos de variables, estos incluyen

puntos.entre.palabras
guiones_bajos_entre_palabras
MayúsculasMinúsculasParaSepararPalabras

También es posible utilizar el operador = para la asignación:

Algunas cosas importantes

- Para R no es lo mismo si las letras son mayúsculas o minúsculas. A no es lo mismo que a
- R no toma en cuenta los espacios en blanco
- R nos puede avisar cuando falta información
- Se pueden agregar comentarios con #
- Help

Vectorización

R es vectorizado, lo que significa que las variables y funciones pueden tener vectores como valores

```
1:5
```

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

```
x <- 1:5
```

```
x <- 1:5
2^x
```

```
## [1]  2  4  8 16 32
```

Vectorización

puedo crear vectores para almacenar muchos números con la función “c”

```
vector<-c(23,45,67,55,34)
vector
```

```
## [1] 23 45 67 55 34
```

```
vector2<-c(12,56,76,34,23)
vector2
```

```
## [1] 12 56 76 34 23
```

```
vector3<-vector+vector2
vector3
```

```
## [1]  35 101 143  89  57
```

Administrando tu entorno

comandos útiles que puedes usar para interactuar con la sesión de R.

```
ls()
```

```
## [1] "A"       "B"       "vector"  "vector2" "vector3" "x"       "y"
```

Puedes usar rm para eliminar objetos que ya no necesitas:

```
rm(x)
```

Paquetes en R

- Es posible agregar funciones a R escribiendo un paquete u obteniendo un paquete escrito por otra persona.
- Hay más de 10,000 paquetes disponibles en CRAN (la red completa de archivos R).
- R y RStudio tienen funcionalidad para administrar paquetes
- Los paquetes se intalan una vez pero se cargan en cada sesión

installed.packages() Ver que paquetes están instalados **install.packages("nombre_de_paquete")** Instalar paquetes **remove.packages("nombre_de_paquete")** eliminar paquetes **library(nombre_de_paquete)** cargar paquetes

Desafío

¿Cuál será el valor de cada variable después de cada comando en el siguiente programa?

mass <- 47.5
age <- 122
mass <- mass * 2.3
age <- age - 20

Ejecuta el código del desafío anterior y escribe un comando para comparar la variable mass con age. ¿Es la variable mass más grande que age?

Limpia tu entorno de trabajo borrando las variables de mass y age.

Instalar el paquete factomineR

Estructura de datos

Una de las características más poderosas de R es su habilidad para manejar datos tabulares.

Comencemos creando un dataset llamado gatos que se vea así.

color	peso	gusto
mixto	2.1	1
negro	5.0	0
atigrado	3.2	1

Estructura de datos

```
gatos <- data.frame(color = c("mixto", "negro", "atigrado"),
                    peso = c(2.1, 5.0, 3.2),
                    le_gusta_cuerda = c(1, 0, 1))
gatos
```

```
##      color peso le_gusta_cuerda
## 1    mixto  2.1               1
## 2    negro  5.0               0
## 3 atigrado  3.2               1
```

Cómo importo datos de excel?

Para acceder a los datos de cada columna

```
gatos$color
```

```
## [1] "mixto"    "negro"    "atigrado"
```

Estructura de datos. Data Frames

Podemos hacer otras operaciones sobre las columnas. Por ejemplo, podemos aumentar el peso de todos los gatos con:

Puedo realizar operaciones

```
gatos$peso + 2
```

```
## [1] 4.1 7.0 5.2
```

Podemos imprimir los resultados en una oración

```
paste("El color del gato es", gatos$color)
```

```
## [1] "El color del gato es mixto"    "El color del gato es negro"
## [3] "El color del gato es atigrado"
```

Data frames

```
gatos[1, 1]
```

```
## [1] "mixto"
```

```
gatos[, 1]
```

```
## [1] "mixto"    "negro"    "atigrado"
```

```
gatos[1, ]
```

```
##   color peso le_gusta_cuerda
## 1 mixto  2.1               1
```

Tipo de datos

Que paso si trato de correr

```
#gatos$peso + gatos$color
```

Error non-numeric argument to binary operator

Hay 5 tipos de datos principales:
- double
- integer
- complex
- logical
- character

tipos de datos

```
class(gatos$color)
```

```
## [1] "character"
```

```
class(gatos$pesos)
```

```
## [1] "NULL"
```

```
str(gatos)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ color          : chr  "mixto" "negro" "atigrado"
##  $ peso           : num  2.1 5 3.2
##  $ le_gusta_cuerda: num  1 0 1
```

Estructura de datos. Matrices

Podemos declarar una matriz llena de ceros de la siguiente forma:

```
matrix_example <- matrix(0, ncol=6, nrow=3)
matrix_example
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    0    0    0    0    0    0
## [3,]    0    0    0    0    0    0
```

Matrices

Podemos preguntar varias cosas de la matrices

```
class(matrix_example)
```

```
## [1] "matrix" "array"
```

```
typeof(matrix_example)
```

```
## [1] "double"
```

```
dim(matrix_example)
```

```
## [1] 3 6
```

```
nrow(matrix_example)
```

```
## [1] 3
```

```
ncol(matrix_example)
```

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

Desafío

Considera la salida de R para la siguiente matriz:

```
 [,1] [,2]
```

[1,] 4 1 [2,] 9 5 [3,] 10 7

¿Cuál fué el comando correcto para escribir esta matriz? Examina cada comando e intenta determinar el correcto antes de escribirlos. Piensa en qué matrices producirán los otros comandos.

1. matrix(c(4, 1, 9, 5, 10, 7), nrow = 3)
2. matrix(c(4, 9, 10, 1, 5, 7), ncol = 2, byrow = TRUE)
3. matrix(c(4, 9, 10, 1, 5, 7), nrow = 2)
4. matrix(c(4, 1, 9, 5, 10, 7), ncol = 2, byrow = TRUE)

Donde seguimos?

https://swcarpentry.github.io/r-novice-gapminder-es/

R para ciencia de datos https://es.r4ds.hadley.nz/

ggplot https://ggplot2-book.org/