

Introducción al análisis de datos con R

Conceptos básicos de R y RStudio

Manuel Mejías Leiva

Universidad de Valladolid | manuel.mejias@uva.es

5 - 9 junio de 2023

Empezando con R y RStudio

¿Qué es R?

R es un **lenguaje estadístico**, creado por y para la estadística, con 4 ventajas fundamentales:

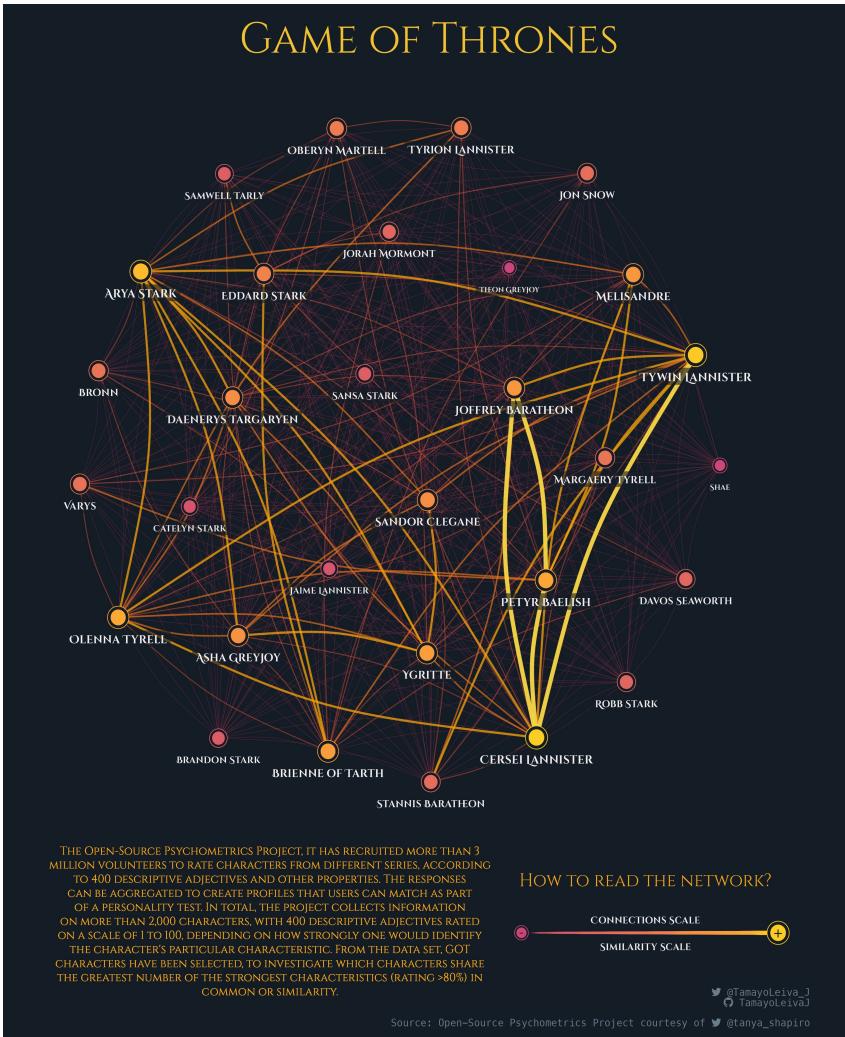
- **Software libre** (como C++, Python, etc). no solo es gratis, sino que permite **acceder libremente a código ajeno**.
- **Lenguaje modular**: al ser software libre, existen **trozos de código** hechos por otras personas (**paquetes**) que podemos instalar según necesidades.
- **Gran comunidad de usuarios**: R tiene una comunidad de usuarios gigante para hacer estadística (Python tiene una comunidad más enfocada al Machine Learning), con más de 18 000 paquetes.
- **Lenguaje de alto nivel**. Los lenguajes de alto nivel, como R o Python, facilitan la programación al usuario (menor curva de aprendizaje, aunque más lentos en ejecución).

¿Por qué utilizar R?

- ¡Es gratis! Si eres profesor/a o estudiante, las ventajas son evidentes.
- Funciona en diversas plataformas, como Windows, Unix y MacOS.
- Proporciona una plataforma inigualable para programar nuevos métodos estadísticos de forma fácil y directa.
- Contiene rutinas estadísticas avanzadas que aún no están disponibles en otros paquetes.
- Dispone de capacidades gráficas de última generación.

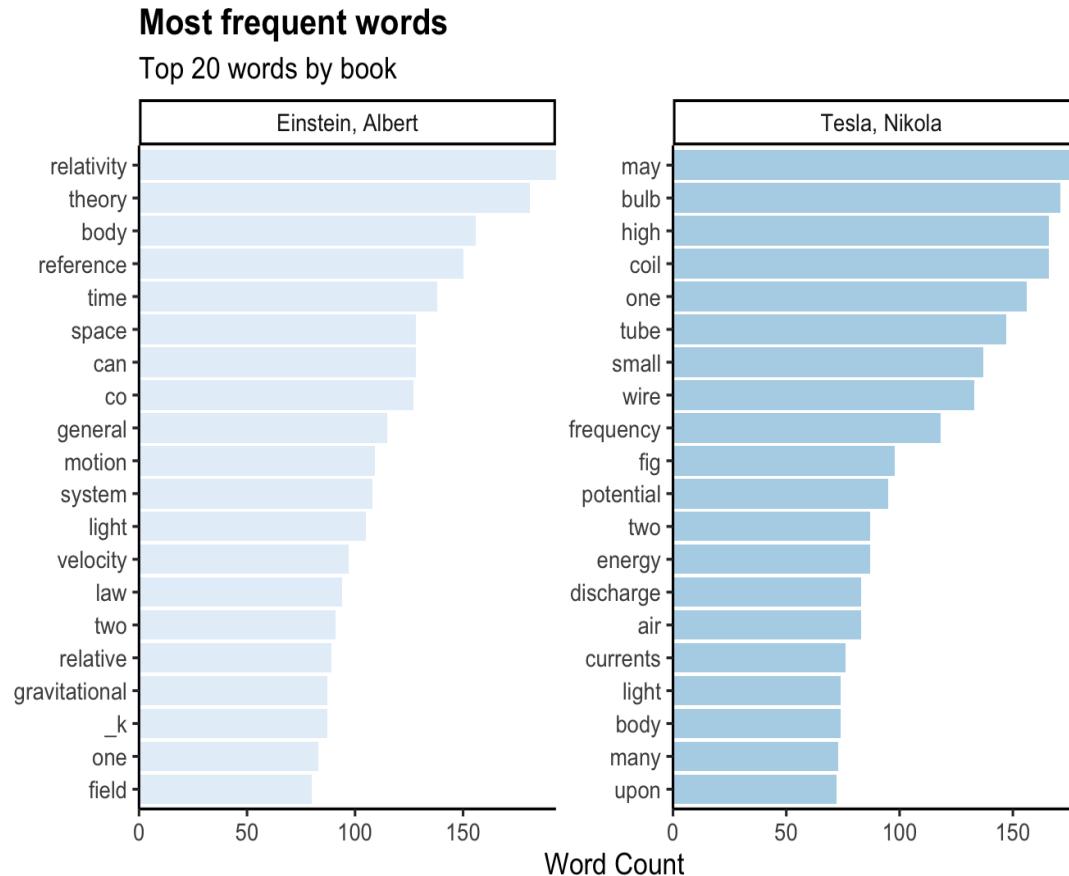
Cosas que se pueden hacer con R

- Análisis de redes



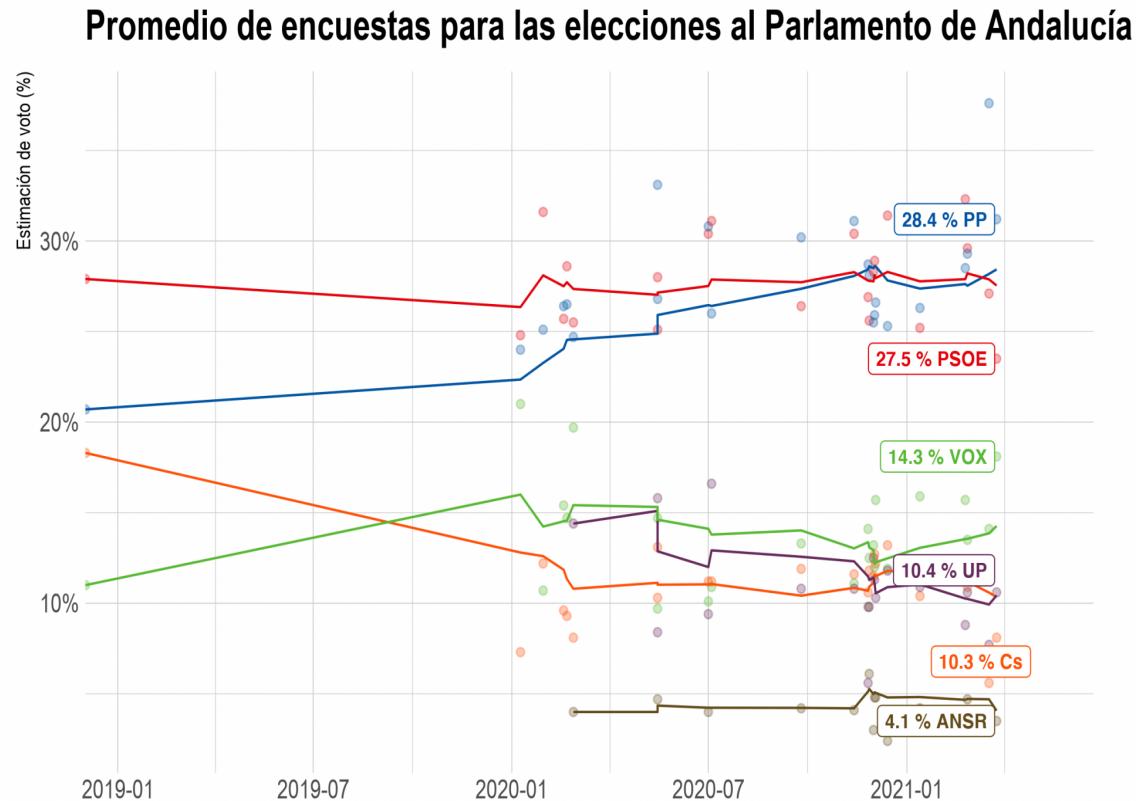
Cosas que se pueden hacer con R

- Análisis de texto



Cosas que se pueden hacer con R

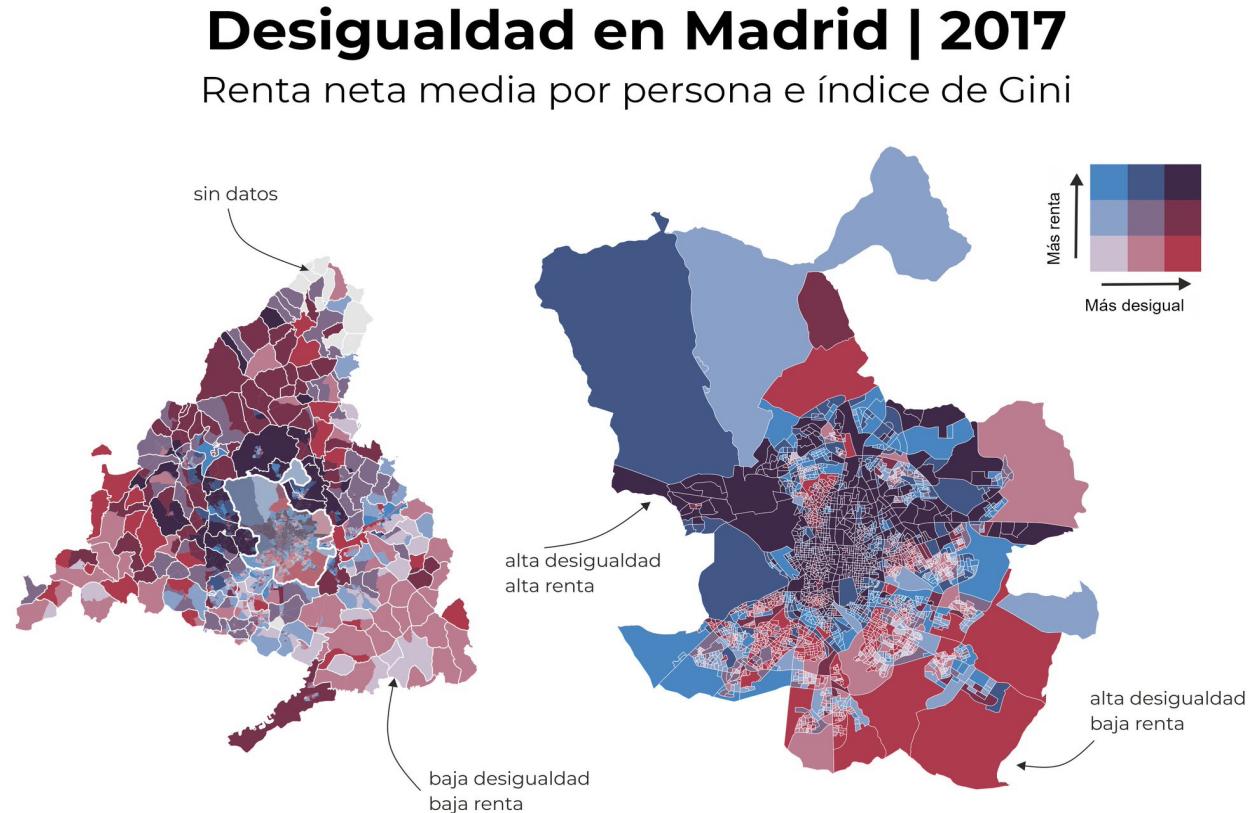
- Web scraping



Manuel Mejías - @MejiasLeiva (Datos: Wikipedia)

Cosas que se pueden hacer con R

- Procesamiento y visualización de datos espaciales



Cosas que se pueden hacer con R

- Creación de aplicaciones interactivas

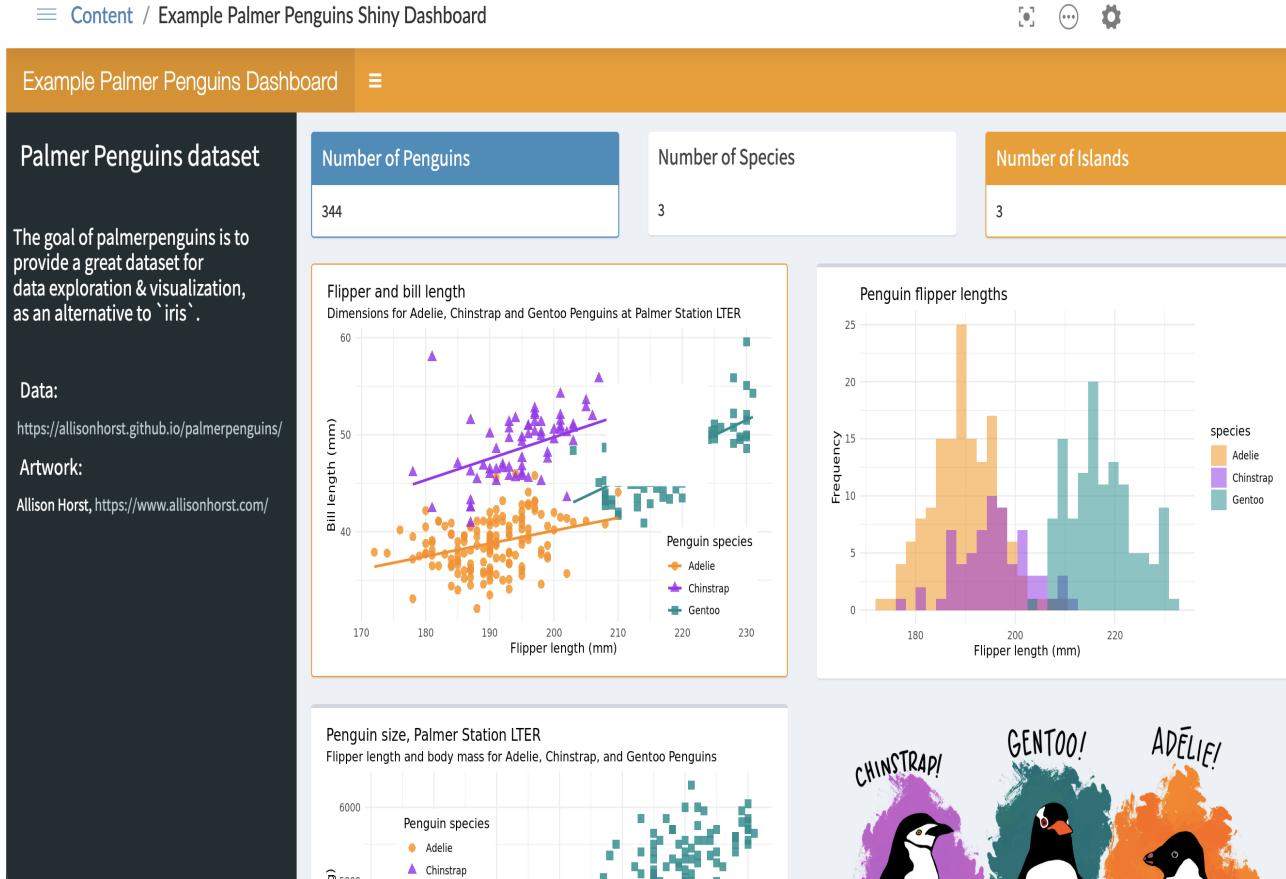


Imagen: Isabella Velásquez

Cosas que se pueden hacer con R

- Presentaciones, informes o artículos académicos
 - Esta misma presentación está hecha en R usando la librería **xaringanthemer**

The image displays two slides from a presentation. The left slide has a yellow background and contains the following text:

Presentation Ninja

⚔️ xaringan +
😎 xaringanthemer

Yihui Xie

Garrick Aden-Buie

2020-04-19

The right slide has a dark blue background and contains the following text:

Hello World

Install the **xaringan** package from [Github](#):

```
devtools::install_github("yihui/xaringan")
```

You are recommended to use the [RStudio IDE](#), but you do not have to.

- Create a new R Markdown document from the menu **File** → **New File** → **R Markdown** → **From Template** → **Ninja Presentation**;¹
- Click the **Knit** button to compile it;
- or use the [RStudio Addin](#)² "Infinite Moon Reader" to live preview the slides (every time you update and save the Rmd document, the slides will be automatically reloaded in RStudio Viewer).

2 / 2

Descargar R y RStudio

Para comenzar a utilizar R, es necesario descargar e instalar tanto el lenguaje de programación como un entorno de desarrollo integrado como RStudio.

Descargar **R**:

- <https://www.r-project.org/>

Descargar **RStudio**:

- <https://www.rstudio.com/products/rstudio/download/>

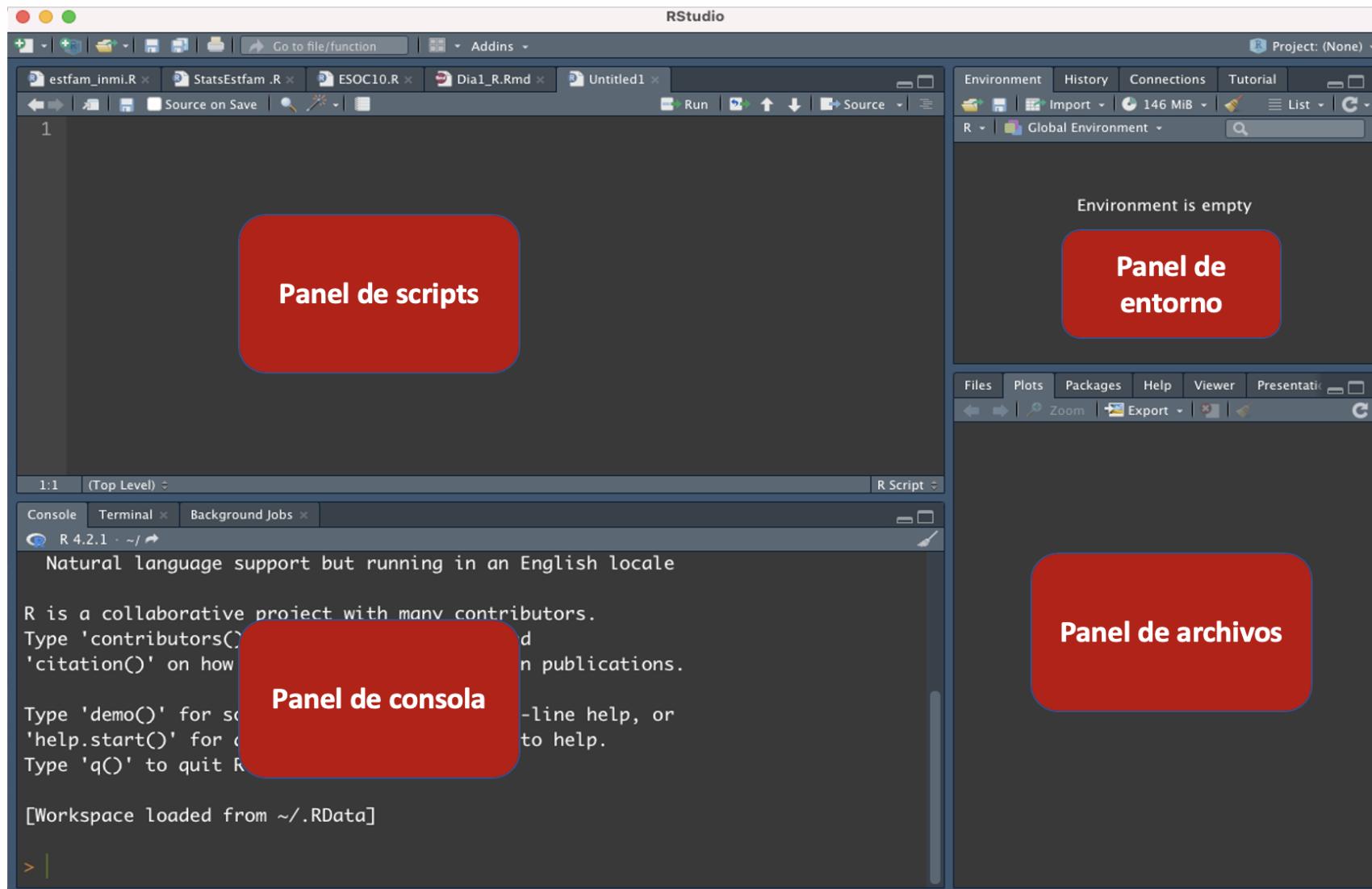
Para instalar R / RStudio, vea este **video**.

Estructura de RStudio

RStudio es un entorno de desarrollo integrado que proporciona una interfaz gráfica de usuario para facilitar el uso de R. La interfaz de RStudio se divide en cuatro paneles principales:

- El panel de **scripts**: donde se escribe y ejecuta el código de R.
- El panel de **consola**: donde se muestra la salida del código que se ejecuta.
- El panel de **entorno**: donde se pueden ver los objetos cargados en R y sus atributos.
- El panel de **archivos**: donde se pueden ver los gráficos, archivos, etc.

Estructura de RStudio



Consejos generales

Antes de empezar, consejos generales

- Empezar un script (`.R`)
 - Limpiar el espacio
 - Cargar paquetes que van a ser usados

```
# Curso R - Sesión 1 -----  
rm(list = ls()) # limpiar el espacio  
  
library(tidyverse) # carga paquete tidyverse  
library(janitor) # carga paquete janitor
```

- Nombrar objetos
 - 2Cs: Conciso y consistente
 - Minúscula
 - Estilos: `sexoEdad`, `sexo_edad` o `sexo.edad`

Antes de empezar, consejos generales

- Comentar `#` [Ctrl + Mayús + C]
 - Explicar el por qué
 - Usar separadores para organizar el scrip [Ctrl + Mayús + R]

```
# Librerías -----  
library(tidyverse)  
  
# Directorio de trabajo -----  
  
# Importar datos -----
```

- Espacios
 - Usar espacios en torno a los operadores

```
a<-22+23 # mal  
a <- 22 + 23 # bien
```

Help!

- Ayuda con las funciones:

```
help(mean)  
?mean
```

Recursos de ayuda a la programación en R:

- [**Stackoverflow**](#)
- [**ChatGPT**](#)
- [**Cheatsheets**](#)
- [**Quick**](#)

Tipos de variables

Tipos de variables

- **Numérica continua:** son variables que pueden tomar cualquier valor dentro de un rango determinado.
 - Ejemplo: Ingreso mensual de los hogares.
- **Numérica discreta:** son variables que sólo pueden tomar valores enteros, es decir, números sin decimales.
 - Ejemplo: Número de hijos en una familia.
- **Categórica nominal:** son variables que describen una cualidad o categoría sin un orden o jerarquía específica.
 - Ejemplo: Género (masculino/femenino).
- **Categórica ordinal:** son variables que describen una cualidad o categoría con un orden o jerarquía específica.
 - Ejemplo: Nivel socioeconómico (bajo/medio/alto).
- **Carácter:** son variables que representan cadenas de caracteres o textos.
 - Ejemplo: Respuestas abiertas en una encuesta.

Primeras operaciones con datos

Primeros pasos en R: calculadora

Usando R como una calculadora:

```
2 * 3
```

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

En R, los **objetos** o **variables** se definen usando el operador `<-`.

```
x <- 5  
y <- x - 4
```

```
y
```

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

Todas las instrucciones en R en las que crees objetos, es decir, las instrucciones de asignación, tienen la misma estructura:

- nombre_objeto `<-` valor

Vectores y data.frame

- En R, un **vector** es una colección ordenada de elementos del mismo tipo. Estos pueden ser **numeric, integers, character, factors o logical**.

```
numeric_vector <- c(1,2,3,4,5)
character_vector <- c("a", "b", "c", "d", "e")
```

- La mayoría de las veces trabajaremos con muchos vectores a la vez, reunidos en un **data frame** o un **tibble**.

```
my_df <- data.frame(fruta = c("manzana", "pera", "uva"),
                     precio = c(2.99, 1.99, 3.99))
```

- No todo en R puede almacenarse en un marco de datos, a veces trabajarás con **lists**.
- Funciones útiles para convertir vectores: **as.numeric()**, **as.character()**, **as.factor()**, etc. Trabajaremos con ellas más adelante.

Seleccionando elementos de un data.frame

¿Cómo seleccionar y filtrar datos?

En el caso de los data.frame tenemos además a nuestro disposición una herramienta muy potente: la función **subset()**.

```
subset(my_df, subset = precio > 2,  
       select = "fruta")
```

```
##      fruta  
## 1 manzana  
## 3 uva
```

Primeros ejercicios



Primeros ejercicios

-  **Ejercicio 1:** añade debajo otra línea para definir una variable **b** con el valor **5**. Tras asignarles valores, multiplica los números en consola.

```
a <- 2
```

Primeros ejercicios

-  **Ejercicio 1:** añade debajo otra línea para definir una variable **b** con el valor **5**. Tras asignarles valores, multiplica los números en consola.

```
a <- 2
```

- Solución ej. 1:

```
# Para poner comentarios en el código se usa #

# Definición de variables
a <- 2
b <- 5

# Multiplicación
a * b
```

```
## [1] 10
```

Primeros ejercicios

-  **Ejercicio 2:** modifica el código inferior para definir dos variables **c** y **d**, con valores 3 y -1, y calcular la división **c/d**

```
c <- # deberías asignarle el valor 3  
d <- # deberías asignarle el valor -1
```

Primeros ejercicios

-  **Ejercicio 2:** modifica el código inferior para definir dos variables **c** y **d**, con valores 3 y -1, y calcular la división **c/d**

```
c <- # deberías asignarle el valor 3  
d <- # deberías asignarle el valor -1
```

- Solución ej. 2:

```
# Definición de variables  
c <- 3  
d <- -1  
# Operación (división)  
c/d
```

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

Primeros ejercicios

-  **Ejercicio 3:** en este ejercicio, tenemos dos vectores: primer_vector, que contiene valores numéricos del 1 al 5, y segundo_vector, que contiene valores de caracteres "a" a "e". Utilizando la función `data.frame()`, creamos un nuevo dataframe llamado nuevo_df.

```
primer_vector <- c(1,2,3,4,5)
segundo_vector <- c("a","b","c","d","e")
```

Primeros ejercicios

-  **Ejercicio 3:** en este ejercicio, tenemos dos vectores: primer_vector, que contiene valores numéricos del 1 al 5, y segundo_vector, que contiene valores de caracteres "a" a "e". Utilizando la función `data.frame()`, creamos un nuevo dataframe llamado nuevo_df.

```
primer_vector <- c(1,2,3,4,5)
segundo_vector <- c("a","b","c","d","e")
```

- Solución ej. 3:

```
nuevo_df <- data.frame(primer_vector,segundo_vector)
nuevo_df
```

```
##   primer_vector segundo_vector
## 1              1                  a
## 2              2                  b
## 3              3                  c
## 4              4                  d
## 5              5                  e
```

Primeros ejercicios

-  **Ejercicio 4:** a partir del data.frame `nuevo_df`, selecciona las filas donde los valores en la columna `primer_vector` sean mayores que 3 utilizando la función `subset()` y asígnalo a un objeto llamado `subset_df`.

Primeros ejercicios

-  **Ejercicio 4:** a partir del data.frame **nuevo_df**, selecciona las filas donde los valores en la columna primer_vector sean mayores que 3 utilizando la función **subset()** y asígnalo a un objeto llamado **subset_df**.
- Solución ej. 4:

```
subset_df <- subset(nuevo_df, primer_vector > 3)
subset_df
```

```
##   primer_vector segundo_vector
## 4                 4                  d
## 5                 5                  e
```

Paquetes, librerías y datos

Paquetes y librerías para importar datos

- **Paquete: haven**

- Archivos: Stata, SPSS, SAS
- Funciones:

- `read_dta("tuarchivo.dta")`
- `read_sav()`
- `read_sas()`

- **Paquete: readr**

- Archivos: csv
- Funciones:

- `read_csv("tuarchivo.csv")`

- **Paquete: readxl**

- Archivos: xls / xlsx
- Funciones:

- `read_xls("tuarchivo.xls")`
- `read_xlsx("tuarchivo.xlsx")`

Paquetes y librerías para exportar datos

Algunas funciones importantes en R para exportar datos:

- `write_excel()` se utiliza para exportar datos a un archivo de Excel (.xlsx).
- `write_sav()` se utiliza para exportar datos a un archivo de formato SPSS (.sav).
- `write_dta()` se utiliza para exportar datos a un archivo de formato Stata (.dta).

Paquetes y librerías

Primero, instalar paquetes

- Un paquete en R es una colección de funciones, datos y documentación que se pueden usar para realizar tareas específicas.

```
# install.packages("tidyverse")
```

Segundo, cargar librerías

- Las librerías son directorios en tu sistema de archivos donde se almacenan los paquetes instalados.

```
library(tidyverse)
```

Directorio de trabajo e importar datos

Antes de cargar los datos debemos de definir nuestro directorio de trabajo. Esto es, debemos indicarle a R cuál es la ubicación en tu ordenador donde se guardan y cargan los archivos de datos y scripts.

```
#setwd("~/Desktop/Intro-R/Dia1")
```

Para saber el directorio de trabajo actual en R, puedes utilizar la función `getwd()`. Por ejemplo:

```
getwd()
```

```
## [1] "/Users/manuelmejiasleiva/Desktop/Intro-R/Dia1"
```

Una vez definido el directorio de trabajo, procedemos a importar los datos a R:

```
data <- read_csv("GSOEP9402.csv")
```

Cambiando la estructura del data frame

Los comandos más comunes que pueden ayudar a modificar el formato de las columnas:

- `as.numeric()` cambiará su formato a numérico.
- `as.character()` cambiará su formato a carácter.
- `as.factor()` cambiará los datos como factores.

```
data$school <- as.factor(data$school)
data$income <- as.numeric(data$income)
```

Conoce tus datos



`head()` muestra las 6 primeras filas de sus datos.

Para ver más (o menos), puedes hacer `head(data, n)`.

```
head(data)
```

```
## # A tibble: 6 × 13
##   ...1 school      birthyear gender  kids parity income size state marital
##   <dbl> <fct>       <dbl> <chr>   <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1     1 Gymnasium    1981 female    2     2 35160.    4 Berlin married
## 2     2 Gymnasium    1981 female    2     2 65748.    3 Berlin married
## 3     3 Gymnasium    1980 female    3     3 120962.   3 Berlin married
## 4     4 Gymnasium    1984 female    1     1 60101.    3 Berlin married
## 5     5 Realschule   1982 male     4     4 34829.    4 Berlin divorced
## 6     6 Realschule   1980 female    3     1 42584.    5 Berlin married
## # i 3 more variables: meducation <dbl>, memployment <chr>, year <dbl>
```

Conoce tus datos

`str()` te da una visión general de los tipos de datos.

```
str(data)
```

```
## #> #> spc_tbl_ [675 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## #> #> $ ...1      : num [1:675] 1 2 3 4 5 6 7 8 9 10 ...
## #> #> $ school    : Factor w/ 3 levels "Gymnasium","Hauptschule",...: 1 1 1 1 3 3 1 1 1 2 ...
## #> #> $ birthyear  : num [1:675] 1981 1981 1980 1984 1982 ...
## #> #> $ gender     : chr [1:675] "female" "female" "female" "female" ...
## #> #> $ kids       : num [1:675] 2 2 3 1 4 3 3 1 4 2 ...
## #> #> $ parity     : num [1:675] 2 2 3 1 4 1 3 1 4 2 ...
## #> #> $ income     : num [1:675] 35160 65748 120962 60101 34829 ...
## #> #> $ size       : num [1:675] 4 3 3 3 4 5 3 3 5 4 ...
## #> #> $ state      : chr [1:675] "Berlin" "Berlin" "Berlin" "Berlin" ...
## #> #> $ marital    : chr [1:675] "married" "married" "married" "married" ...
## #> #> $ meducation : num [1:675] 14.5 10.5 12 10.5 10 15 15 13 15 9 ...
## #> #> $ memployment: chr [1:675] "none" "parttime" "parttime" "parttime" ...
## #> #> $ year       : num [1:675] 1995 1995 1994 1998 1996 ...
## #> #> - attr(*, "spec")=
## #> #>   .. cols(
## #> #>     .. ...1 = col_double(),
## #> #>     .. school = col_character().
```

Conoce tus datos

`glimpse()` es útil para obtener una vista previa rápida de la estructura de un dataframe y el tipo de datos de sus variables.

```
glimpse(data)
```

```
## Rows: 675
## Columns: 13
## $ ...1      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
## $ school    <fct> Gymnasium, Gymnasium, Gymnasium, Gymnasium, Realschule, Re...
## $ birthyear <dbl> 1981, 1981, 1980, 1984, 1982, 1980, 1986, 1986, 1981, 1983...
## $ gender    <chr> "female", "female", "female", "female", "male", "female", ...
## $ kids       <dbl> 2, 2, 3, 1, 4, 3, 3, 1, 4, 2, 2, 2, 5, 5, 5, 2, 1, 2, 4, 4...
## $ parity    <dbl> 2, 2, 3, 1, 4, 1, 3, 1, 4, 2, 1, 1, 3, 2, 1, 2, 1, 2, 1, 3...
## $ income    <dbl> 35160.11, 65748.35, 120962.36, 60100.57, 34828.95, 42584.0...
## $ size       <dbl> 4, 3, 3, 3, 4, 5, 3, 3, 5, 4, 4, 4, 7, 7, 7, 4, 3, 4, 6, 4...
## $ state     <chr> "Berlin", "Berlin", "Berlin", "Berlin", "Berlin", "Berlin"...
## $ marital   <chr> "married", "married", "married", "married", "divorced", "m...
## $ meducation <dbl> 14.5, 10.5, 12.0, 10.5, 10.0, 15.0, 15.0, 13.0, 15.0, 9.0...
## $ memployment <chr> "none", "parttime", "parttime", "parttime", "fulltime", "p...
## $ year      <dbl> 1995, 1995, 1994, 1998, 1996, 1994, 2000, 2000, 1995, 1997...
```

Conoce tus datos

- La función `summary()` se utiliza para obtener un resumen estadístico de un objeto, como un vector, una matriz o un marco de datos. El resumen proporciona información como el mínimo, el primer cuartil, la mediana, el tercer cuartil y el máximo de los datos, así como el número de valores faltantes.

```
summary(data)
```

```
##      ...1          school      birthyear      gender
## Min.   : 1.0   Gymnasium :277   Min.   :1980   Length:675
## 1st Qu.:169.5  Hauptschule:199  1st Qu.:1982   Class  :character
## Median :338.0  Realschule :199  Median :1984   Mode   :character
## Mean   :338.0                               Mean   :1984
## 3rd Qu.:506.5                               3rd Qu.:1986
## Max.  :675.0                               Max.  :1988
##      kids          parity      income          size
## Min.   :1.00   Min.   :1.00   Min.   : 1248   Min.   :2.000
## 1st Qu.:2.00   1st Qu.:1.00   1st Qu.: 49229  1st Qu.:4.000
## Median :2.00   Median :2.00   Median : 66555  Median :4.000
## Mean   :2.51   Mean   :1.76   Mean   : 71311  Mean   :4.259
## 3rd Qu.:3.00   3rd Qu.:2.00   3rd Qu.: 86646  3rd Qu.:5.000
## Max.  :6.00   Max.   :5.00   Max.   :258341  Max.   :8.000
##      state         marital      meducation      memployment
## Min.   :1.00   Min.   :1.00   Min.   : 1248   Min.   :2.000
## 1st Qu.:2.00   1st Qu.:1.00   1st Qu.: 49229  1st Qu.:4.000
## Median :2.00   Median :2.00   Median : 66555  Median :4.000
## Mean   :2.51   Mean   :1.76   Mean   : 71311  Mean   :4.259
## 3rd Qu.:3.00   3rd Qu.:2.00   3rd Qu.: 86646  3rd Qu.:5.000
## Max.  :6.00   Max.   :5.00   Max.   :258341  Max.   :8.000
```

Conoce tus datos



`summary()` le proporciona estadísticas resumidas de sus datos. Puede aplicarlo a todo el data frame o **a una sola variable**.

```
summary(data$income)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max. 
##    1248    49229   66555    71311   86646  258341
```

```
summary(data$meducation)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max. 
##    7.00   10.50  11.00    11.44   12.00  18.00
```

- El símbolo del dólar (\$) se utiliza en la programación en R para **seleccionar una columna de un data frame**.

Conoce tus datos

- Contar los valores perdidos en cada columna

```
colSums(is.na(data))
```

```
##      ...1    school birthyear     gender      kids    parity
##          0        0        0        0        0        0        0
## income      size      state marital meducation memployment
##          0        0        0        0        0        0        0
## year      0
```

Conoce tus datos

`table()` cuenta el número de veces que aparece cada categoría y muestra los resultados en una tabla.

```
table(data$school)
```

```
##  
##   Gymnasium  Hauptschule  Realschule  
##      277        199        199
```

También es posible crear tablas que incluyan dos o más variables categóricas:

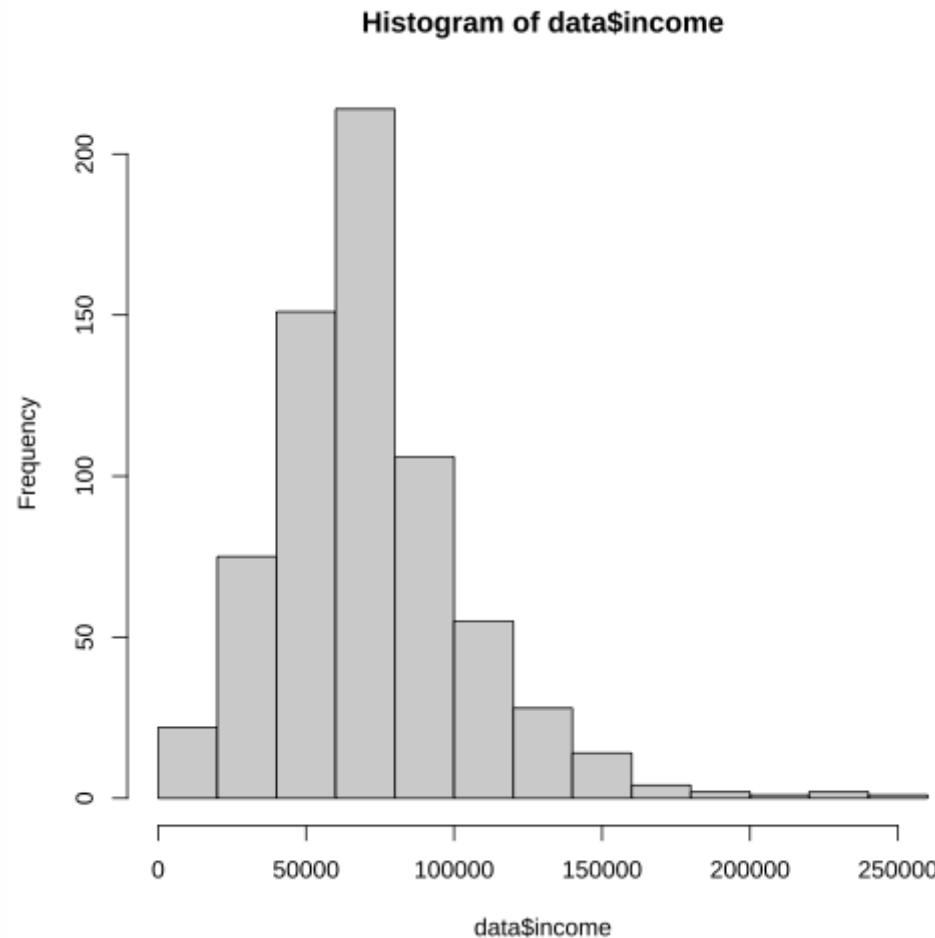
```
table(data$school,data$marital)
```

```
##  
##                      divorced married separated single widowed  
##   Gymnasium           11     245       13      2       6  
##   Hauptschule         23     164       7       4       1  
##   Realschule          25     157       9       5       3
```

Conoce tus datos

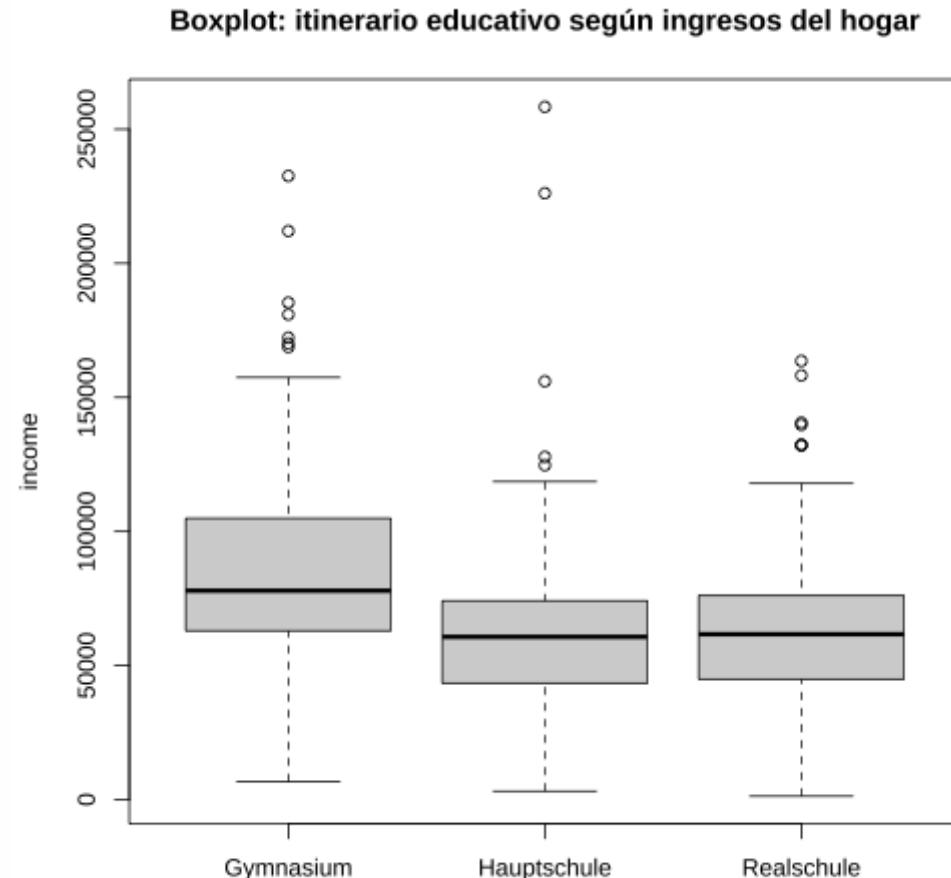


```
hist(data$income)
```



Conoce tus datos

```
boxplot(income ~ school, data = data,  
        main = "Boxplot: itinerario educativo según ingresos del hogar")
```



Conoce tus datos



Calcular la media de ingresos para el conjunto de la muestra:

```
mean(data$income)
```

```
## [1] 71310.96
```

Mediana de ingresos para el conjunto de la muestra:

```
median(data$income)
```

```
## [1] 66555.03
```

Conoce tus datos



Encontrar el valor de renta del hogar más bajo del conjunto de datos:

```
min(data$income)
```

```
## [1] 1247.59
```

Encontrar el valor de renta del hogar más alto del conjunto de datos:

```
max(data$income)
```

```
## [1] 258341.4
```

Conoce tus datos



Para calcular la media por grupos se puede utilizar la función **aggregate**. Por ejemplo, vamos a calcular la media de años de educación de las madres agrupada por el itinerario educativo seguido por los hijos/as:

```
aggregate(meducation ~ school, data = data, mean)
```

```
##           school meducation
## 1   Gymnasium    12.55415
## 2 Hauptschule    10.33668
## 3  Realschule    11.00251
```

Conoce tus datos



- Para analizar variables categóricas:

```
library(janitor)
```

```
tabyl(data$school)
```

```
##   data$school    n    percent
##   Gymnasium 277 0.4103704
##   Hauptschule 199 0.2948148
##   Realschule 199 0.2948148
```

Conoce tus datos



```
data %>%  
  tabyl(school,gender)
```

```
##          school female male  
##   Gymnasium      151  126  
##   Hauptschule     96   103  
##   Realschule     101   98
```

- Aquí hay un nuevo signo extraño: `%>%`. Esta cosa se llama "**pipe**". Puedes leerlo como decirle a R: "**primero haz esto y luego haz aquello**". Así, primero le dices a R que use el objeto "data" y luego (%>%) le dices a R que tome ese objeto y realice la operación `tabyl()` con ese objeto.
- También podéis encontrar este signo `|>` que es la última actualización del **pipe** en R.

Conoce tus datos

```
data %>%
  tabyl(school,gender) %>%
  adorn_percentages("row")
```

```
##      school    female     male
##      Gymnasium 0.5451264 0.4548736
##      Hauptschule 0.4824121 0.5175879
##      Realschule 0.5075377 0.4924623
```

Conoce tus datos

```
data %>%
  tabyl(school,gender) %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 1)
```

```
##      school female male
##    Gymnasium 54.5% 45.5%
##  Hauptschule 48.2% 51.8%
##  Realschule 50.8% 49.2%
```

Guardar script



- File >> Save As...
- Elige una ubicación en tu ordenador donde deseas guardar el script.
- Asigna un nombre descriptivo al archivo. Por convención, los scripts de R suelen tener una extensión **.R** al final del nombre del archivo (por ejemplo, **mi_script.R**).

Recursos



Recursos para aprender

- Libros: **R for Data Science**
- Cursos web: **DataCamp**
- Twitter: **RLadiesGlobal, hadleywickham, dataandme, andrewheiss**
- Rbloggers: : <https://www.r-bloggers.com/>