ULPGC
Universidad de
Las Palmas de
Gran Canaria
Departamento de
Análisis Económico Aplicado

EITT

Introducción al análisis de datos con R

José Manuel Cazorla Artiles

Estructura del curso

1. Descarga e instalación de R y RStudio.
2. Primeros pasos con R.
    2.1. Tipos de objetos: vectores, matrices, data frames y listas.
    2.2. Estructuras básicas de programación.
    2.3. Trabajo con paquetes.
3. Tratamiento de datos.
    3.1. Importar y exportar datos.
    3.2. Preparación de datos.
    3.3. Caso práctico: trabajar con microdatos.
4. Visualización de datos con ggplot2.
5. Informes con RMarkdown.
6. Ampliar conocimientos.

Descarga e instalación de R y RStudio

¿Qué es R?

R es un entorno y lenguaje de programación diseñado para el análisis estadístico.

- Es gratuito y de código abierto. Es parte de un proyecto colaborativo y abierto donde, además de la configuración básica, hay multitud de funciones y paquetes desarrollados por los propios usuarios.
- Al ser un software de código abierto, a diferencia de otras herramías de análisis estadístico como STATA, SPSS, MATLAB... no tiene restricciones a la modificación del código, salvo las propias de la licencia GPL.

¿Y RStudio?

RStudio es un entorno de desarrollo integrado (IDE en su siglas en inglés) para R.

- Incluye consola, editor con resaltado de sintaxis que soporta ejecución directa del código además de herramientas para gestión de gráficos, historial de comandos, depuración de código y espacio de trabajo.
- Es de código abierto y existen versiones para los diferentes sistemas operativos (Windows, Mac y Linux).

Descarga e instalación

R

- Web de R
- Link de descarga

RStudio

- Web de RStudio
- Link de descarga

Primeros pasos en R

En su versión más básica R funciona como una calculadora.

Operadores aritméticos

- Suma: `+`
- Resta: `-`
- Multiplicación:`*`
- División:`/`
- Potencia: `^`
- Módulo: `%%`

Pruebe el siguiente código en la consola:

```
3 + 4
```

Variables

Un concepto fundamental en los lenguajes de programación es el de **variable**.

Permite almacenar un valor o un objeto en R para luego poder consultarlo o emplearlo como parte de otras operaciones.

Ejemplo

Para asignar el valor `4` a la variable `x` se puede usar el operador asignación `<-` (se puede usar también `=`)

```
x <- 4
```

Para acceder al valor almacenado en `x` debemos escribir el nombre de la variable en la consola

```
x
```

R nos devuelve en consola el valor almacenado en `x` que en este caso es `4`

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

Tipos de datos en R

- Texto (character) `chr: "char"`
- Numérico `num: 3.4`
- Entero `int: 3L`
- Complejo `cplx: 3i`
- Lógico `logi: TRUE`
- Fecha `Date`: as.Date("2020-01-01")

Podemos consultar el tipo de objeto mediante la función `class(x)`.

Ejemplo

Aplique la función class(x) a los objetos definidos a continuación:

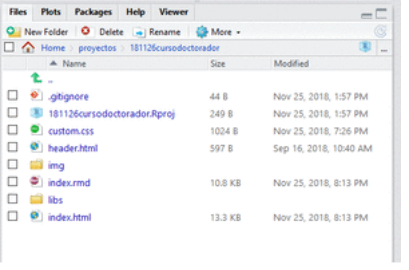
```
numerico <- 3.4
entero <- 3L
complejo <- 3i
logico <- TRUE
fecha <- as.Date("2020-01-01")
```

```
class(x = numerico)
```

```
## [1] "numeric"
```

Directorio de trabajo

- Vía menú
  - Ir a la pestaña `Files`
  - Hacer click en `...` (Go to directory)
  - Seleccionar directorio
  - Hacer click en la subestaña con la rueda dentada `More`
  - Hacer click en `Set AS Working Directory`



- Vía comando de R

```
setwd('c:/establecer/carpeta/de/trabajo')
```

- Vía proyecto
  - `File`, `New Project`. Cuando esté creado se puede acceder al proyecto con `File`, `Open Project`.

Tipos de objetos

Vectores

Los vectores son conjuntos de datos unidimensionales del mismo tipo. Se declaran con la función `c()` (de concatenate).

Ejemplo

Crear vectores numérico, lógico y de texto y comprobar sus clases.

```
vnum <- c(1,2,3)
vlog <- c(TRUE,FALSE,TRUE)
vchar <- c("curso","de","R")
```

```
class(vnum)
```

```
## [1] "numeric"
```

```
class(vlog)
```

```
## [1] "logical"
```

```
class(vchar)
```

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

Puede obtenerse la longitud de un vector con la función `length(x)`.

```
length(vnum)
```

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

Operaciones con vectores

- Concatenar vectores

```
vec1 <- c(1,4,5,3)
vec2 <- 1:4 # es equivalente a c(1,2,3,4) o seq(1,4,1)
vec12 <- c(vec1, vec2)
```

```
vec12
```

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

- Operaciones con un escalar

```
vec1 + 2
```

```
## [1] 3 6 7 5
```

```
vec1 * 2
```

```
## [1]  2  8 10  6
```

- Operaciones vectoriales

```
vec1 + vec2
```

```
## [1] 2 6 8 7
```

Operaciones con vectores

Comparadores

- Mayor que: >
- Menor que: <
- Mayor o igual que: >=
- Menor o igual que: <=
- Igual: ==
- Distinto: !=

- Comparar vectores

```
vec1 < vec2
```

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

- Concatenar el vector de texto `vchar` y los valores de `vec1`

```
paste(vchar, vec1)
```

```
## [1] "curso 1" "de 4"    "R 5"     "curso 3"
```

- Concatenar los valores de `vchar` en un único string

```
paste(vchar, collapse = " ")
```

```
## [1] "curso de R"
```

Seleccionar elementos de vectores

Tomemos un nuevo vector de ejemplo.

```
vec1 <- 1:7
```

- Seleccionar un elemento

```
vec1[1]
```

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

- Seleccionar elementos por nombre

```
# asignamos un nombre a cada elemento del vector
names(vec1) <- c("lunes", "martes", "miercoles",
                 "jueves", "viernes", "sabado", "domingo")
vec1[c("miercoles")]
```

```
## miercoles
##         3
```

- Seleccionar elementos que cumplen una condición

```
vec1[vec1 < 5]
```

```
##     lunes    martes miercoles    jueves
##         1         2         3         4
```

Matrices

La matriz es un objeto bidimensional de un mismo tipo de datos repartidos en filas y columnas.

Se puede crear una matriz usando la función `matrix(x)`.

Por ejemplo, una matriz de 3x3 con valores de 1 a 9 ordenados por fila.

```
matriz1 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = TRUE)
# Ordenados por columnas
# matriz1 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = FALSE)
matriz1
```

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

```
class(matriz1)
```

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

Puede omitirse el argumento de columnas/filas pues la función `matrix(x)` es capaz de obtenerlo a partir del número de elemento si conoce al menos una dimensión.

```
matriz1 <- matrix(data = 1:9, nrow = 3, byrow = TRUE)
matriz1
```

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

Puede obtenerse la dimensón de una matriz con la función `dim(x)`.

```
dim(matriz1)
```

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

Añadir filas y columnas mediante las funciones `rbind(x)` y `cbind(x)`.

```
rbind(matriz1, 1:3)
```

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

```
cbind(matriz1, 1:3)
```

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

Las filas y columnas de una matriz pueden nombrarse mediante las funciones `rownames(x)` y `colnames(x)`.

```
rownames(matriz1) <- paste("fila", 1:3)
colnames(matriz1) <- paste("columna", 1:3)
matriz1
```

```
##        columna 1 columna 2 columna 3
## fila 1         1         2         3
## fila 2         4         5         6
## fila 3         7         8         9
```

Operaciones con matrices

Las operaciones con matrices siguen la misma lógica que con los vectores. Veamos algunos ejemplo:

- Suma de un escalar

```
matriz1 + 2
```

```
##        columna 1 columna 2 columna 3
## fila 1         3         4         5
## fila 2         6         7         8
## fila 3         9        10        11
```

- Producto de los elementos de una matriz

```
matriz1 * matriz1
```

```
##        columna 1 columna 2 columna 3
## fila 1         1         4         9
## fila 2        16        25        36
## fila 3        49        64        81
```

- Producto de matrices

```
matriz1 %*% matriz1
```

```
##        columna 1 columna 2 columna 3
## fila 1        30        36        42
## fila 2        66        81        96
## fila 3       102       126       150
```

Operaciones con matrices

- Suma por fila

```
rowSums(matriz1)
```

```
## fila 1 fila 2 fila 3
##      6     15     24
```

- Suma por columna

```
colSums(matriz1)
```

```
## columna 1 columna 2 columna 3
##        12        15        18
```

- Producto por fila

```
apply(matriz1, 1, prod)
```

```
## fila 1 fila 2 fila 3
##      6    120    504
```

- Producto por columna

```
apply(matriz1, 2, prod)
```

```
## columna 1 columna 2 columna 3
##        28        80       162
```

Seleccionar elementos de matrices

A diferencia de los vectores, donde sólo necesitamos indicar un valor pues son unidimensionales, en las matrices hay que indicar dos valores ya que tienen dos dimensions.

- Posición de fila y columna

```
matriz1[2,3]
```

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

```
matriz1[2,1:3]
```

```
## columna 1 columna 2 columna 3
##         4         5         6
```

- Nombre de fila y columna

```
matriz1["fila 1", "columna 1"]
```

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

```
matriz1["fila 1", paste("columna", 1:2)]
```

```
## columna 1 columna 2
##         1         2
```

Factores

Se emplean para almacenar variables categóricas en R y pueden ser factores ordenados o no ordenados.

Se crean con la función `factor(x)`.

- Factor no ordenado

```
sexo <- factor(c("Hombre", "Mujer", "Hombre", "Mujer"))
sexo
```

```
## [1] Hombre Mujer  Hombre Mujer
## Levels: Hombre Mujer
```

- Factor ordenado

```
estudios <- factor(c("primarios", "secundarios", "superiores",
                     "superiores", "secundarios", "primarios"),
                   ordered = TRUE,
                   levels = c("primarios", "secundarios", "superiores"))
estudios
```

```
## [1] primarios   secundarios superiores  superiores  secundarios primarios
## Levels: primarios < secundarios < superiores
```

Permite la comparación entre los elementos de dicho factor.

```
estudios[1] > estudios[2]
```

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

Listas

La lista es un objeto que permite recoger objetos de distinto tipo.

Se puede crear una lista con la función `list()`.

```
lista1 <- list(vector = vec1, matriz = matriz1,
               factor.no.ordenado = sexo, factor.ordenado = estudios)
str(lista1) # muestra la estructura
```

```
## List of 4
##  $ vector            : Named int [1:7] 1 2 3 4 5 6 7
##   ..- attr(*, "names")= chr [1:7] "lunes" "martes" "miercoles" "jueves" ...
##  $ matriz            : int [1:3, 1:3] 1 4 7 2 5 8 3 6 9
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "fila 1" "fila 2" "fila 3"
##   .. ..$ : chr [1:3] "columna 1" "columna 2" "columna 3"
##  $ factor.no.ordenado: Factor w/ 2 levels "Hombre","Mujer": 1 2 1 2
##  $ factor.ordenado   : Ord.factor w/ 3 levels "primarios"<"secundarios"<..: 1 2 3 3 2 1
```

Seleccionar elementos de una lista

Los elementos de una lista se puden seleccionar con el operador `[[]]`, o `$` si el elemento tiene nombre.

```
lista1$vector
```

```
##     lunes    martes miercoles    jueves   viernes    sabado   domingo
##         1         2         3         4         5         6         7
```

```
lista1[[1]]
```

```
##     lunes    martes miercoles    jueves   viernes    sabado   domingo
##         1         2         3         4         5         6         7
```

```
lista1[["vector"]]
```

```
##     lunes    martes miercoles    jueves   viernes    sabado   domingo
##         1         2         3         4         5         6         7
```

También se pueden seleccionar elementos dentro de un elmento mediante el operador `[]`.

```
lista1$vector[1]
```

```
## lunes
##     1
```

```
lista1$vector[1:2]
```

```
##  lunes martes
##      1      2
```

Operaciones con listas

Ejemplos

- Suma de un escalar.

```
lista1$vector + 1
```

```
##     lunes    martes miercoles    jueves   viernes    sabado   domingo
##         2         3         4         5         6         7         8
```

- Suma por columna de una matriz.

```
colSums(lista1$matriz)
```

```
## columna 1 columna 2 columna 3
##        12        15        18
```

Data Frame

Se trata de un objeto de caracter bidimensional, en el que cada columna sí debe ser del mismo tipo de datos.

Se crean con la función `data.frame()`.

```
id <- 1:5
ola <- c(rep("T1",3), rep("T2", 2))
sexo <- factor(c('Hombre', 'Mujer', 'Mujer', 'Mujer', 'Hombre'))
ingresos <- c(1500,2300,1700,900,2100)
residente <- c(TRUE, TRUE, TRUE, FALSE, TRUE)
isla <- c("Gran Canaria","Tenerife", "Tenerife", NA, "Gran Canaria")

encuesta <- data.frame(id, ola, sexo, ingresos, residente, isla, stringsAsFactors = FALSE)
encuesta
```

```
##   id ola   sexo ingresos residente         isla
## 1  1  T1 Hombre     1500      TRUE Gran Canaria
## 2  2  T1  Mujer     2300      TRUE     Tenerife
## 3  3  T1  Mujer     1700      TRUE     Tenerife
## 4  4  T2  Mujer      900     FALSE         <NA>
## 5  5  T2 Hombre     2100      TRUE Gran Canaria
```

Obtener la estructura del dataframe.

```
str(encuesta)
```

```
## 'data.frame':    5 obs. of  6 variables:
##  $ id       : int  1 2 3 4 5
##  $ ola      : chr  "T1" "T1" "T1" "T2" ...
##  $ sexo     : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 2 1
##  $ ingresos : num  1500 2300 1700 900 2100
```

Seleccionar elementos de un dataframe

La selección de elementos se realiza con el operador `[]`, también podemos seleccionar una columna por su nombre con el operador `$`.

```
# encuesta["sexo"]
encuesta$sexo
```

```
## [1] Hombre Mujer  Mujer  Mujer  Hombre
## Levels: Hombre Mujer
```

```
# encuesta[c(1,2,3)]
encuesta[c("sexo", "ingresos", "residente")]
```

```
##     sexo ingresos residente
## 1 Hombre     1500      TRUE
## 2  Mujer     2300      TRUE
## 3  Mujer     1700      TRUE
## 4  Mujer      900     FALSE
## 5 Hombre     2100      TRUE
```

```
encuesta[1:2, 2:3]
```

```
##   ola   sexo
## 1  T1 Hombre
## 2  T1  Mujer
```

```
encuesta[encuesta$ingresos > 2000, ]
```

```
##   id ola   sexo ingresos residente         isla
## 2  2  T1  Mujer     2300      TRUE     Tenerife
## 5  5  T2 Hombre     2100      TRUE Gran Canaria
```

Operaciones con dataframes

Por ejemplo, crear un nuevo dataframe reemplazando la variable `ingresos` por `ingresos * 2`.

```
ingresos2 <- encuesta$ingresos * 2
ingresos2
```

```
## [1] 3000 4600 3400 1800 4200
```

```
encuesta2 <- encuesta
encuesta2$ingresos <- ingresos2
encuesta2
```

```
##   id ola   sexo ingresos residente         isla
## 1  1  T1 Hombre     3000      TRUE Gran Canaria
## 2  2  T1  Mujer     4600      TRUE     Tenerife
## 3  3  T1  Mujer     3400      TRUE     Tenerife
## 4  4  T2  Mujer     1800     FALSE         <NA>
## 5  5  T2 Hombre     4200      TRUE Gran Canaria
```

Estructuras básicas de programación

Condicionales

Se puede controlar el flujo de ejecución mediante el uso de condicionales.

```
x <- -1
if(x < 0) {
  print("El valor de x es negativo")
}
```

```
## [1] "El valor de x es negativo"
```

```
x <- 1
if(x < 0) {
  print("El valor de x es negativo")
} else {
  print("El valor de x es positivo")
}
```

```
## [1] "El valor de x es positivo"
```

```
x <- 0
if(x < 0) {
  print("El valor de x es negativo")
} else if (x > 0) {
  print("El valor de x es positivo")
} else {
  print("El valor de x es 0")
}
```

```
## [1] "El valor de x es 0"
```

Bucles

Permite realizar un conjunto de operaciones de manera iterativa.

```
x <- 0
for (i in 1:10) {
  x <- x + i
  print(x)
}
```

```
## [1] 1
## [1] 3
## [1] 6
## [1] 10
## [1] 15
## [1] 21
## [1] 28
## [1] 36
## [1] 45
## [1] 55
```

```
x <- 0
while (x < 5) {
  x <- x + 1
  print(x)
}
```

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

Funciones

Las funciones son conjuntos de insrtucciones que realizan tareas determinadas y que pueden depender de parámetros que modifican el resultado.

Ejemplos

```
f_suma <- function(x, y) {x + y}
f_suma(1, 2)
```

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

```
f_suma_positivos <- function(x, y) {
  if(any(c(x,y) < 0)) {
    print("Introduzca valores positivos")
    } else {
      x + y
    }
}


f_suma_positivos(1, -2)
```

```
## [1] "Introduzca valores positivos"
```

Trabajo con paquetes

Una de las virtudes de R es la ingente cantidad de funciones que hay desarrolladas.

Los usuarios de R desarrollan sus propias funciones y pueden agrupar éstas en un paquete.

En CRAN, el repositorio oficial de R hay numerosos paquetes con distintas utilidades.

Estos paquetes nos facilitan la tarea de la programación. Para poder usarlos debemos instalarlos y posteriormente cargarlos.

Instalar un paquete de R

- Con el comando `install.package()`.
- A través de la pestaña `Packages` y pulsando el botón `Install`.
- A través del menú `Tools`, `Install packages`.

Cargar un paquete de R

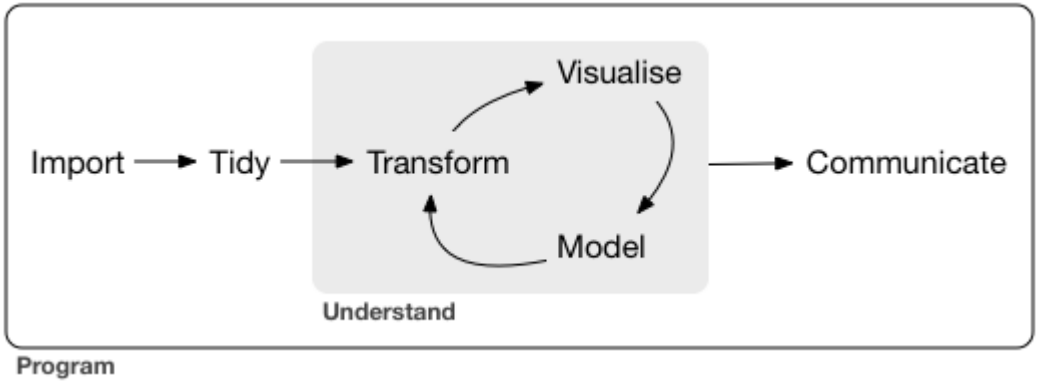
Con el comando `library()`.

- Pulsando en la casilla de verificación de la lista de paquetes intalados en la pestaña de `Packages`.

Tratamiento de datos

El flujo de trabajo

El flujo de trabajo en un proceso de análisis de datos puede representarse con el siguiente diagrama en la mayoría de los casos.



Fuente: R for Data Science

- **Import**. El primer paso consiste en acceder a los datos. Los datos pueden estar en multitud de formatos: desde EXCEL y CSV hasta datos web, imágenes etc.
- **Tidy**. Posteriormente se realiza un proceso de limpieza de los datos. Los datos no siempre tienen un formato amigable, de hecho, lo normal es que no lo tengan, si lo tienen es que alguien ha hecho este paso previamente.
- **Understand**. Una vez los datos estén en el formato adecuado se inicia el proceso de exploración de los datos. En esta parte del flujo de trabajo se realizan transformaciones y visualizaciones de los datos con objeto de entender mejor los mismos y las relaciones entre ellos, normalmente es un proceso iterativo. La modelización forma parte del proceso de análisis de datos aunque queda fuera del alcance de este curso.
- **Communicate**. Consiste en reportar, de la manera adecuada, las conclusiones extraidas.

Importar y exportar datos

Importar fichero CSV

Leer el fichero `serie_gasto_istac.csv`, que contiene las series trimestrales de gasto turístico. Islas de Canarias. 2018 (Metodología 2018) según la EGT recogidas en el ISTAC, de la carpeta `data` y almacenarlo en la variable `datos`. Mostrar las 5 primeras filas.

```
library(readr)

datos <- read_csv2("data/serie_gasto_istac.csv")
head(datos)
```

```
## # A tibble: 6 x 13
##   Periodo    TOTAL   Alemania  Bélgica  España  Francia  Holanda Irlanda Italia
##   <chr>      <chr>   <chr>     <chr>    <chr>   <chr>    <chr>   <chr>   <chr>
## 1 2021 Prim~ 477659~ 15260651~ 1270554~ 780309~ 5853140~ 501033~ 811012~ 193948~
## 2 2020       486783~ 10870007~ 1948477~ 573726~ 2093751~ 177528~ 119279~ 133156~
## 3 2020 Cuar~ 666689~ 17762377~ 4540235~ 103904~ 3584793~ 125273~ 164285~ 146798~
## 4 2020 Terc~ 803894~ 17080218~ 5000075~ 254588~ 4402044~ 358075~ 151699~ 332754~
## 5 2020 Segu~ ..       ..        ..       ..      ..       ..      ..      ..
## 6 2020 Prim~ 339724~ 738574784 9944465~ 215233~ 1295067~ 129193~ 876813~ 852009~
## # ... with 4 more variables: Países Nórdicos <chr>, Reino Unido <chr>,
## #   Suiza <chr>, Otros países <chr>
```

También se puede importar desde el menú mediante el botón `Import Dataset` de la pestaña `Environment`.

Importar fichero excel

Leer el fichero `serie_gasto_istac.xlsx` con los datos anteriormente cargados en formato CSV, ahora tienen formato XLSX. Mostrar las 5 primeras filas.

```
library(readxl)

datos <- read_excel("data/serie_gasto_istac.xlsx")
head(datos)
```

```
## # A tibble: 6 x 13
##   Periodo   TOTAL   Alemania Bélgica  España  Francia  Holanda  Irlanda  Italia
##   <chr>     <chr>   <chr>    <chr>    <chr>   <chr>    <chr>    <chr>    <chr>
## 1 2021 Pri~ 477659~ 1526065~ 1270554~ 780309~ 5853140~ 5010331~ 8110127~ 193948~
## 2 2020      486783~ 1087000~ 1948477~ 573726~ 2093751~ 1775284~ 1192798~ 133156~
## 3 2020 Cua~ 666689~ 1776237~ 4540235~ 103904~ 3584793~ 1252735~ 1642854~ 146798~
## 4 2020 Ter~ 803894~ 1708021~ 5000075~ 254588~ 4402044~ 3580756~ 1516992~ 332754~
## 5 2020 Seg~ ..      ..       ..       ..      ..       ..       ..       ..
## 6 2020 Pri~ 339724~ 7385747~ 9944465~ 215233~ 1295067~ 1291934~ 8768139~ 852009~
## # ... with 4 more variables: Países Nórdicos <chr>, Reino Unido <chr>,
## #   Suiza <chr>, Otros países <chr>
```

APIs

Un API "Application Programming Interface" es un mecanismo que permite realizar una petición de datos a un servidor de internet.

Muchas instituciones prestan un servicio API para consultar los datos que ofrecen.

Permiten un acceso programático a los datos, que se obtienen, frecuentemente, en formato JSON.

Es habitual el desarrollo de paquetes en R que faciliten el acceso a estos datos.

Destacar algunas iniciativas de datos en abierto como: ROpenSci y ROpenSpain.

A continuación se indican algunos de ellos:

Paquete	Fuente de datos	Código de instalación
`WDI`	Banco Mundial	`install.packages("WDI")`
`imfr`	Fondo Monetario Internacional	`install.packages("imfr")`
`eurostat`	Eurostat	`install.packages("eurostat")`
`ecb`	Banco Central Europeo	`install.packages("ecb")`
`tidyBDE`	Banco de España	`install.packages("tidyBDE")`
`INEbaseR`	Instituto Nacional de Estadística	`remotes::install_github("oddworldng/INEbaseR")`
`istacr`	Instituto Canario de Estadística	`install.packages("istacr")`
`istacbaser`	Instituto Canario de Estadística	`remotes::install_github("ropenspain/istacbaser")`
`igebaser`	Instituto Galego de Estatística	`remotes::install_github("jmcartiles/igebaser")`
`opendataes`	Iniciativa datos.gob.es	`remotes::install_github("ropenspain/opendataes")`
...		

Otros formatos de datos

- Mediante el botón `Import Dataset` de la pestaña `Environment`
- Mediante el paquete `haven`.

```
install.packages("haven")
dataset <- read_stata() # Stata
dataset <- read_sav() # SPSS
dataset <- read_sas() # SAS
```

Exportar datos

Hay varias funciones que permiten grabar y más tarde cargar los ficheros ya grabados.

	Guardar	**Leer**
Todas	`save.image()`	`load()`
Algunos objetos	`save()`	`load()`
Un objeto	`saveRDS()`, `readr::write_rds()`	`readRDS()`, `readr::read_rds()`
Csv	`write.csv()`, `readr:write_csv()`	`read.csv()`, `readr::read_csv()`
Excel	`xlsx::write.xlsx()`	`readr::read_excel()`

Preparación de datos

Una vez cargados los datos, comienza el proceso de procesamiento de los mismos, suele decirse que ocupa en torno al 80% del tiempo del proyecto.

En este curso realizaremos esta tarea mediante el conjunto de paquetes de tidyverse.

The pipe

El operador pipe (`%>%`), puede leerse como "entonces" y permite encadenar funciones pasando el elemento que está a la izquierda del pipe a la derecha como un argumento.

```
library(tidyverse)
```

```
data.frame(resultado = mean(1:4))
```

```
##   resultado
## 1       2.5
```

```
1:4 %>%
  mean %>%
  data.frame(resultado = .)
```

```
##   resultado
## 1       2.5
```

DPLYR

Es un paquete de R, que forma parte del entorno tidyverse, y nos facilita la manipulación de datos.

Principales funciones

- `filter()`: seleccionar filas
- `arrange()`: ordenar filas
- `rename()`: renombrar variables
- `select()`: seleccionar columnas
- `mutate()`: crear variables
- `summarise()`: resume varios valores en uno
- `group_by()`: agrupar filas

A partir del dataframe `encuesta` aplicar las funciones del paquete `dplyr` definidas anteriormente.

- Seleccionar a los encuestados que declaran ser residentes en Canarias.

```
encuesta.filter <- filter(encuesta, residente == TRUE)
```

- Ordenar de manera descendente los ingresos.

```
encuesta.arrange <- arrange(encuesta, desc(ingresos))
```

- Renombrar la variable `isla` como `isla de residencia`

```
encuesta.rename <- rename(encuesta, "isla de residencia" = "isla")
```

- Seleccionar las columnas: `sexo`, `ingresos` e `isla`

```
encuesta.select <- select(encuesta, c("sexo", "ingresos", "isla"))
```

- Crear la variable `ccaa` que indica la comunidad autónoma de residencia.

```
encuesta.mutate <- mutate(encuesta, ccaa = ifelse(residente == TRUE, "Canarias", "Otra CCAA"))
```

- Resumir el dataframe a una varible que indique el ingreso medio.

```
encuesta.summarise <- summarise(encuesta, ingreso.medio = mean(ingresos))
```

- Calcular el ingreso medio por isla de residencia.

```
encuesta.group_by <- summarise(group_by(encuesta, isla), mean(ingresos))
```

Aplique las funciones anteriores empleando el operador `%>%`.

Ejercicios

- `filter`
    - Seleccionar los encuestados que son hombres.
    - Seleccionar los encuestados que son mujeres.
    - Seleccionar los encuestados con ingresos inferiores a 1000.
    - Seleccionar los encuestados con ingresos superiores a 1000.
    - Seleccionar los encuestados no residentes.
    - Seleccionar los encuestados residentes en Gran Canaria.
    - Seleccionar los encuestados residentes en Tenerife.
- `arrange`
    - Ordenar la encuesta según la variable de tipo factor `sexo`.
    - Ordenar la encuesta según la variable de tipo numérico `ingresos`.
    - Ordenar la encuesta según la variable de tipo lógico `residente`.
    - Ordenar la encuesta según la variable de tipo caracter `isla`.
    - Realizar las ordenaciones anteriores de forma descendente.
- `select`
    - Seleccionar todas las variables menos `sexo`.
    - Seleccionar todas las variables menos `sexo` e `ingresos`.
    - Seleccionar todas las variables que contengan la letra `e`. (Pista: función `contains()`)
    - Seleccionar todas las variables que no contengan la letra `e`. (Pista: función `contains()`)
    - Seleccionar las variables que empiecen con la letra `s`. (Pista: función `starts_with()`)
    - Seleccionar las variables que terminen con la letra `o`. (Pista: función `end_with()`)

- `mutate`
    - Crear la variable `ingresos.2` como los ingresos al cuadrado.
    - Crear la variable `ingresos.2.res` como los ingresos al cuadrado para los residentes. (Pista: función `case_when()`)
    - Crear la variable `ingresos.cat` que toma valor 0 si `ingresos < 1000`, 1 si `ingresos >= 1000 & ingresos < 2000` y 2 si `ingresos >= 2000`. (Pista: función `case_when()`)
    - Reemplazar en la variable `isla` los na por "Otra CCAA". (Pista: función `replace_na()`)
- `summarise`
    - Resumir el dataframe en la variable `ingreso.total` que indica la suma de ingreso de los encuestados.
    - Resumir el dataframe en la variable `ingreso.dt` que indica la desviación típica del ingreso de los encuestados.
    - Resumir el dataframe en la variable `resindentes.media` que indica la cuota de residentes entre los encuestados.
- `group_by`
    - Crear la variable `ingreso.medio.isla` que indica el promedio de ingresos por isla de residencia.
    - Crear la variable `ingreso.total.isla` que indica el promedio de ingresos por isla de residencia.
    - Crear la variable `encuestados.isla` que indica el total de encuestados por isla de residencia. (Pista: función `n()`)
    - Crear la variable `encuestados.freq.isla` que indica la cuota de encuestados por isla de residencia. (Pista: función `n()`)

De ancho a largo y de largo a ancho

En ciencias sociales los datos suelen tener una estructura tabular, es decir, se organizan en filas y columnas. Donde cada fila indica una unidad de análisis y cada columna una variable.

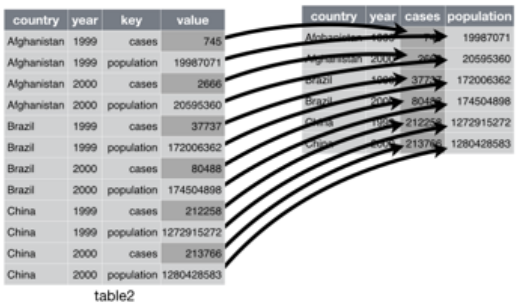
Es preciso indicar que las variables se pueden agrupar en variables de identificación y de medida.

Así, podemos distinguir entre datos en formato ancho, cuando cada columna es una variable, y formato largo cuando cada fila es una combinación de variables de identificación.

Para trabajar con las funciones tidyverse es más útil tener los datos en formato largo.

Pero en otras ocasiones puede ser de mayor interés tener los datos en formato ancho, veamos como cambiar de uno a otro.

De largo a ancho



De ancho a largo

- Dataframe de ejemplo:

```
df.largo <- data.frame(
  poblacion = c(1000, 2000, 3000, 4000, 5000, 6000),
  region = c("A", "A", "A", "B", "B", "B"),
  periodo = c(rep(seq(1,3,1), 2))
)
```

- De largo a ancho.

```
df.ancho <- pivot_wider(df.largo, id_cols = region,
            names_from = "periodo", names_prefix = "periodo_", values_from = "poblacion")
df.ancho
```

```
## # A tibble: 2 x 4
##   region periodo_1 periodo_2 periodo_3
##   <chr>      <dbl>     <dbl>     <dbl>
## 1 A           1000      2000      3000
## 2 B           4000      5000      6000
```

- De ancho a largo.

```
df.largo <- pivot_longer(df.ancho,
                         cols = c(starts_with("periodo")), names_to = "periodo",
                         names_prefix = "periodo_", values_to = "poblacion")
df.largo
```

```
## # A tibble: 6 x 3
##   region periodo poblacion
##   <chr>  <chr>       <dbl>
## 1 A      1            1000
## 2 A      2            2000
## 3 A      3            3000
## 4 B      1            4000
## 5 B      2            5000
## 6 B      3            6000
```

Caso práctico: trabajar con microdatos

Descargar los microdatos de 2019 de la Encuesta de Gasto Turístico del ISTAC, según la metodología 2018.

- Cargar los microdatos.

```
df.microdatos.2019 <- read_csv2("data/GASTO_TURISTICO_2019.csv")
```

- Mostrar la estructura de los datos.

```
glimpse(df.microdatos.2019)
```

```
## Rows: 38,761
## Columns: 168
## $ ID                                <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12~
## $ NUMERO_CUESTIONARIO               <dbl> 1, 10, 100, 101, 102, 103, 104, 105, ~
## $ OLA                               <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ AEROPUERTO_ORIGEN                 <chr> "ACE", "ACE", "ACE", "ACE", "ACE", "A~
## $ PAIS_DESTINO                      <chr> "FRA250", "DEU276", "IRL372", "IRL372~
## $ SEXO                              <dbl> 1, 1, 1, 1, 1, 6, 1, 6, 6, 6, 1, 6, 1~
## $ EDAD                              <dbl> 67, 40, 51, 60, 69, 24, 49, 47, 31, 7~
## $ NACIONALIDAD                      <chr> "FRA250", "DEU276", "IRL372", "IRL372~
## $ PAIS_RESIDENCIA                   <chr> "FRA250", "DEU276", "IRL372", "IRL372~
## $ PROPOSITO                         <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ MOTIVACION_1                      <dbl> 2, 4, 4, 2, 1, 1, 1, 4, 4, 1, 5, 2, 2~
## $ IMPORTANCIA_CLIMA                 <dbl> 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4~
## $ IMPORTANCIA_PLAYAS                <dbl> 4, 4, 4, 2, 4, 4, 2, 4, 3, 3, 2, 2, 3~
## $ IMPORTANCIA_MAR                   <dbl> 4, 4, 3, 2, 2, 3, 3, 4, 3, 3, 4, 2, 4~
## $ IMPORTANCIA_PAISAJES              <dbl> 4, 4, 2, 4, 1, 3, 1, 2, 3, 4, 4, 4, 3~
## $ IMPORTANCIA_ENTORNO_AMBIENTAL     <dbl> 4, 3, 1, 1, 2, 3, 2, 2, 3, 4, 3, 4, 3~
## $ IMPORTANCIA_RED_SENDEROS          <dbl> 3, 1, 1, 1, 1, 1, 1, 2, 1, 1, 3, 3, 2~
## $ IMPORTANCIA_OFERTA_ALOJATIVA      <dbl> 3, 4, 3, 2, 4, 3, 4, 4, 4, 3, 4, 4, 3~
```

- Seleccionar las columnas: `SEXO`, `PAIS_RESIDENCIA`, `EDAD`, `NOCHES`, `PERSONAS_16_64`, `PERSONAS_65_O_MAS`, `GASTO_EUROS`, `COSTE_VUELOS_EUROS`, `COSTE_ALOJ_EUROS`, `ISLA`, `ALOJ_CATEG`, `ANTELACION_VIAJE`, `TRIMESTRE` y `PESO`.

  Para conocer las variables disponibles en el fichero es recomendable mirar el fichero `Diseño de registro EGT 2018.xls`.

```
df.microdatos.2019 <- select(df.microdatos.2019, c("SEXO", "PAIS_RESIDENCIA", "EDAD", "NOCHES",
                                                   "PERSONAS_16_64", "PERSONAS_65_O_MAS",
                                                   "GASTO_EUROS", "COSTE_VUELOS_EUROS",
                                                   "COSTE_ALOJ_EUROS", "ISLA", "ALOJ_CATEG",
                                                   "ANTELACION_VIAJE", "TRIMESTRE", "PESO"))
```

- Mostrar la estructura de los datos.

```
glimpse(df.microdatos.2019)
```

```
## Rows: 38,761
## Columns: 14
## $ SEXO                <dbl> 1, 1, 1, 1, 1, 6, 1, 6, 6, 6, 1, 6, 1, 6, 1, 1, 6, ~
## $ PAIS_RESIDENCIA     <chr> "FRA250", "DEU276", "IRL372", "IRL372", "IRL372", "~
## $ EDAD                <dbl> 67, 40, 51, 60, 69, 24, 49, 47, 31, 70, 57, 67, 72,~
## $ NOCHES              <dbl> 7, 7, 10, 12, 9, 6, 7, 10, 7, 7, 7, 14, 14, 7, 7, 7~
## $ PERSONAS_16_64      <dbl> 1, 2, 2, 2, 0, 2, 2, 2, 2, 3, 2, 0, 0, 2, 1, 2, 0, ~
## $ PERSONAS_65_O_MAS   <dbl> 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 2, ~
## $ GASTO_EUROS         <dbl> 500.0800, 500.1500, 420.0000, 1000.0800, 1000.0800,~
## $ COSTE_VUELOS_EUROS <dbl> 451.0143, 1550.0705, 1800.0000, 999.9600, 750.0214,~
## $ COSTE_ALOJ_EUROS    <dbl> 698.9457, 650.0295, 2296.0000, 999.9600, 650.0186, ~
## $ ISLA                <chr> "ES708", "ES708", "ES708", "ES708", "ES708", "ES708~
## $ ALOJ_CATEG          <dbl> 2, 5, 2, 4, 2, 2, 3, 4, 1, 2, 4, 5, 2, 4, 2, 4, 2, ~
## $ ANTELACION_VIAJE    <dbl> 5, 5, 5, 4, 6, 5, 5, 5, 6, 6, 5, 6, 6, 4, 2, 6, 6, ~
## $ TRIMESTRE           <chr> "2019Q2", "2019Q2", "2019Q2", "2019Q2", "2019Q2", "~
## $ PESO                <dbl> 519.4322, 428.1090, 348.4510, 238.4456, 465.9586, 1~
```

- Reemplazar el código de la variable `SEXO` por la etiqueta correspondiente de acuerdo al diseño de registro.
    - Leer la pestaña `Sexo` del fichero `Diseño de registro EGT 2018.xls`.
        - Eliminar variable `OBSERVACIONES`.
        - Renombrar variables `CÓDIGOS` y `ETIQUETAS` a `code` y `label`.
        - Saltar la primera fila.
        - Eliminar valores NA.
    - Sustituir el código por la etiqueta.
        - Identificar la correspondencia del código de `df.microdatos.2019` en `df.disenyo.registro.sexo`.
        - Reemplazar la variable `SEXO`, que muestra los códigos, por la variable `label`.
        - Transformar la variable `label` de caracter a factor

```
df.disenyo.registro.sexo <- read_excel("data/Diseño de registro EGT 2018.xls",
                                       sheet = "Sexo", skip = 1) %>%
  select("code" = "CÓDIGOS", "label" = "ETIQUETA") %>%
    filter(!is.na(code))
```

```
i.code <- match(df.microdatos.2019$SEXO, df.disenyo.registro.sexo$code)
df.microdatos.2019 <- df.microdatos.2019 %>%
  mutate("SEXO" = as.factor(df.disenyo.registro.sexo$label[i.code]))
```

Aplique el procedimiento anterior para las variables: `PAIS_RESIDENCIA`, `ISLA` y `ALOJ_CATEG`.

- Mostrar la estructura de los datos.

```
glimpse(df.microdatos.2019)
```

```
## Rows: 38,761
## Columns: 14
## $ SEXO                <fct> HOMBRE, HOMBRE, HOMBRE, HOMBRE, HOMBRE, MUJER, HOMB~
## $ PAIS_RESIDENCIA     <chr> "FRA250", "DEU276", "IRL372", "IRL372", "IRL372", "~
## $ EDAD                <dbl> 67, 40, 51, 60, 69, 24, 49, 47, 31, 70, 57, 67, 72,~
## $ NOCHES              <dbl> 7, 7, 10, 12, 9, 6, 7, 10, 7, 7, 7, 14, 14, 7, 7, 7~
## $ PERSONAS_16_64      <dbl> 1, 2, 2, 2, 0, 2, 2, 2, 2, 3, 2, 0, 0, 2, 1, 2, 0, ~
## $ PERSONAS_65_O_MAS   <dbl> 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 2, ~
## $ GASTO_EUROS         <dbl> 500.0800, 500.1500, 420.0000, 1000.0800, 1000.0800,~
## $ COSTE_VUELOS_EUROS <dbl> 451.0143, 1550.0705, 1800.0000, 999.9600, 750.0214,~
## $ COSTE_ALOJ_EUROS    <dbl> 698.9457, 650.0295, 2296.0000, 999.9600, 650.0186, ~
## $ ISLA                <chr> "ES708", "ES708", "ES708", "ES708", "ES708", "ES708~
## $ ALOJ_CATEG          <dbl> 2, 5, 2, 4, 2, 2, 3, 4, 1, 2, 4, 5, 2, 4, 2, 4, 2, ~
## $ ANTELACION_VIAJE    <dbl> 5, 5, 5, 4, 6, 5, 5, 5, 6, 6, 5, 6, 6, 4, 2, 6, 6, ~
## $ TRIMESTRE           <chr> "2019Q2", "2019Q2", "2019Q2", "2019Q2", "2019Q2", "~
## $ PESO                <dbl> 519.4322, 428.1090, 348.4510, 238.4456, 465.9586, 1~
```

- Crear una función para cambiar los códigos por etiquetas.

  Los parámetros de la función serán:

  - `var_variable`. Indica la variable cuyos códigos se van a reemplazar.
  - `var_ruta`. Indica la ruta en la que se encuentra el fichero `Diseño de registro EGT 2018.xls`.
  - `var_sheet`. Indica la hoja de EXCEL en la que se encuentra la relación de código y etiqueta.
  - `var_factor`. Indica si la variable es de tipo caracter o factor, por defecto caracter.

```
get_label <- function(var_variable, var_sheet, var_ruta, var_factor = FALSE) {

  df.disenyo.registro.tmp <- read_excel(var_ruta, sheet = var_sheet, skip = 1) %>%
    select("code" = "CÓDIGOS", "label" = "ETIQUETA") %>%
    filter(!is.na(code))

  i.code <- match(var_variable, df.disenyo.registro.tmp$code)

  v.result <- df.disenyo.registro.tmp$label[i.code]

  if(var_factor == TRUE) v.result <- as.factor(v.result)

  return(v.result)

}
```

- Aplicar la función a la variable `PAIS_RESIDENCIA`.

```
df.microdatos.2019 <- df.microdatos.2019 %>%
  mutate(`PAIS_RESIDENCIA` = get_label(
    var_variable = `PAIS_RESIDENCIA`,
    var_sheet = "País2",
    var_ruta = "data/Diseño de registro EGT 2018.xls")
)
```

- Mostrar la estructura de los datos.

```
glimpse(df.microdatos.2019)
```

```
## Rows: 38,761
## Columns: 14
## $ SEXO                <fct> HOMBRE, HOMBRE, HOMBRE, HOMBRE, HOMBRE, MUJER, HOMB~
## $ PAIS_RESIDENCIA     <chr> "FRANCIA", "ALEMANIA", "IRLANDA", "IRLANDA", "IRLAN~
## $ EDAD                <dbl> 67, 40, 51, 60, 69, 24, 49, 47, 31, 70, 57, 67, 72,~
## $ NOCHES              <dbl> 7, 7, 10, 12, 9, 6, 7, 10, 7, 7, 7, 14, 14, 7, 7, 7~
## $ PERSONAS_16_64      <dbl> 1, 2, 2, 2, 0, 2, 2, 2, 2, 3, 2, 0, 0, 2, 1, 2, 0, ~
## $ PERSONAS_65_O_MAS   <dbl> 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 2, ~
## $ GASTO_EUROS         <dbl> 500.0800, 500.1500, 420.0000, 1000.0800, 1000.0800,~
## $ COSTE_VUELOS_EUROS <dbl> 451.0143, 1550.0705, 1800.0000, 999.9600, 750.0214,~
## $ COSTE_ALOJ_EUROS    <dbl> 698.9457, 650.0295, 2296.0000, 999.9600, 650.0186, ~
## $ ISLA                <chr> "ES708", "ES708", "ES708", "ES708", "ES708", "ES708~
## $ ALOJ_CATEG          <dbl> 2, 5, 2, 4, 2, 2, 3, 4, 1, 2, 4, 5, 2, 4, 2, 4, 2, ~
## $ ANTELACION_VIAJE    <dbl> 5, 5, 5, 4, 6, 5, 5, 5, 6, 6, 5, 6, 6, 4, 2, 6, 6, ~
## $ TRIMESTRE           <chr> "2019Q2", "2019Q2", "2019Q2", "2019Q2", "2019Q2", "~
## $ PESO                <dbl> 519.4322, 428.1090, 348.4510, 238.4456, 465.9586, 1~
```

- Aplicar la función a la variable `ISLA`.

```
df.microdatos.2019 <- df.microdatos.2019 %>%
  mutate(`ISLA` = get_label(
    var_variable = `ISLA`,
    var_sheet = "Isla",
    var_ruta = "data/Diseño de registro EGT 2018.xls")
)
```

- Mostrar la estructura de los datos.

```
glimpse(df.microdatos.2019)
```

```
## Rows: 38,761
## Columns: 14
## $ SEXO                <fct> HOMBRE, HOMBRE, HOMBRE, HOMBRE, HOMBRE, MUJER, HOMB~
## $ PAIS_RESIDENCIA     <chr> "FRANCIA", "ALEMANIA", "IRLANDA", "IRLANDA", "IRLAN~
## $ EDAD                <dbl> 67, 40, 51, 60, 69, 24, 49, 47, 31, 70, 57, 67, 72,~
## $ NOCHES              <dbl> 7, 7, 10, 12, 9, 6, 7, 10, 7, 7, 7, 14, 14, 7, 7, 7~
## $ PERSONAS_16_64      <dbl> 1, 2, 2, 2, 0, 2, 2, 2, 2, 3, 2, 0, 0, 2, 1, 2, 0, ~
## $ PERSONAS_65_O_MAS   <dbl> 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 2, 0, 0, 0, 2, ~
## $ GASTO_EUROS         <dbl> 500.0800, 500.1500, 420.0000, 1000.0800, 1000.0800,~
## $ COSTE_VUELOS_EUROS <dbl> 451.0143, 1550.0705, 1800.0000, 999.9600, 750.0214,~
## $ COSTE_ALOJ_EUROS    <dbl> 698.9457, 650.0295, 2296.0000, 999.9600, 650.0186, ~
## $ ISLA                <chr> "LANZAROTE", "LANZAROTE", "LANZAROTE", "LANZAROTE",~
## $ ALOJ_CATEG          <dbl> 2, 5, 2, 4, 2, 2, 3, 4, 1, 2, 4, 5, 2, 4, 2, 4, 2, ~
## $ ANTELACION_VIAJE    <dbl> 5, 5, 5, 4, 6, 5, 5, 5, 6, 6, 5, 6, 6, 4, 2, 6, 6, ~
## $ TRIMESTRE           <chr> "2019Q2", "2019Q2", "2019Q2", "2019Q2", "2019Q2", "~
## $ PESO                <dbl> 519.4322, 428.1090, 348.4510, 238.4456, 465.9586, 1~
```

> Una variable clave al trabajar con microdatos es el factor de ponderación, nos indica el peso que tiene un encuestado cuando elevamos el resultado de la muestra a la población. En el caso de la EGT, el factor de ponderación es la variable `PESO`.

- Obtener el número de turistas según sexo.
    - Emplear el factor de ponderación `PESO`.
    - Agrupar el dataframe por `SEXO`.
    - Calcular la suma de la variable `PESO` para cada grupo.

```
df.microdatos.2019 %>%
  group_by(SEXO) %>%
  summarise(turistas.mas.16 = sum(PESO)) %>%
  ungroup
```

```
## # A tibble: 2 x 2
##   SEXO   turistas.mas.16
##   <fct>            <dbl>
## 1 HOMBRE        6453123.
## 2 MUJER         6822711.
```

Ejercicio propuesto

Obtener el total de turistas mayores de 16 años.

- Obtener el gasto turístico por isla.
    - Crear la variable `gasto.total` como la suma de `GASTO_EUROS`, `COSTE_VUELOS_EUROS` y `COSTE_ALOJ_EUROS`.
    - Crear la variale `turistas.mas.16` como la suma de `PERSONAS_16_64` y `PERSONAS_65_O_MAS`.
    - Agrupar el dataframe por `ISLA`.
    - Calcular el gasto total por turista mayor de 16 años y emplear el factor de ponderación `PESO`.
    - Calcular la suma de `PESO*GASTO_EUROS` para cada grupo.

```
df.microdatos.2019 %>%
  mutate(gasto.total = GASTO_EUROS + COSTE_VUELOS_EUROS + COSTE_ALOJ_EUROS,
         turistas.mas.16 = PERSONAS_16_64 + PERSONAS_65_O_MAS) %>%
  group_by(ISLA) %>%
  summarise(gasto.total = sum(gasto.total/turistas.mas.16*PESO)) %>%
  ungroup
```

```
## # A tibble: 7 x 2
##   ISLA                gasto.total
##   <chr>                     <dbl>
## 1 EL HIERRO_LA GOMERA   73923565.
## 2 FUERTEVENTURA       1959315660.
## 3 GRAN CANARIA        4341862169.
## 4 LA PALMA             283790289.
## 5 LANZAROTE           2746862301.
## 6 No procede            70548945.
## 7 TENERIFE            5615155360.
```

Ejercicios propuestos

Obtener el gasto turístico total en Canarias.
Obtener el gasto turístico total según el país de residencia.
Obtener el gasto turístico total en cada isla según el país de residencia.

- Obtener los turistas mayores de 16 años y el gasto medio por turista en cada isla.

  El procedimiento es similar al anterior pero dividiendo el gasto total por isla por la cantidad de turistas de cada isla.

```
df.microdatos.2019 %>%
  mutate(gasto.total = GASTO_EUROS + COSTE_VUELOS_EUROS + COSTE_ALOJ_EUROS,
         turistas.mas.16 = PERSONAS_16_64 + PERSONAS_65_O_MAS) %>%
  group_by(ISLA) %>%
  summarise(turistas.mas.16.total = sum(PESO),
            gasto.medio = sum(gasto.total/turistas.mas.16*PESO)/turistas.mas.16.total) %>%
  ungroup
```

```
## # A tibble: 7 x 3
##   ISLA                turistas.mas.16.total gasto.medio
##   <chr>                               <dbl>       <dbl>
## 1 EL HIERRO_LA GOMERA                63118.       1171.
## 2 FUERTEVENTURA                    1659115.       1181.
## 3 GRAN CANARIA                     3702777.       1173.
## 4 LA PALMA                          235559.       1205.
## 5 LANZAROTE                        2521668.       1089.
## 6 No procede                         53214.       1326.
## 7 TENERIFE                         5040382.       1114.
```

Ejercicios propuestos

Obtener el número de turistas y el gasto medio por turista en Canarias.
Obtener el número de turistas y el gasto medio por turista en Canarias según el país de residencia.
Obtener el número de turistas y el gasto medio por turista en cada isla según el país de residencia.

- Obtener los turistas mayores de 16 años y el gasto medio por turista y noche en cada isla.
    - Agrupar el dataframe por islas, calcular para cada encuestado mayor de 16 años su gasto por noche.
    - Elevar el resultado atendiendo al factor de ponderación.
    - Dividir el resultado entre la suma del factor de ponderación.

```
df.microdatos.2019 %>%
  mutate(gasto.total = GASTO_EUROS + COSTE_VUELOS_EUROS + COSTE_ALOJ_EUROS,
         turistas.mas.16 = PERSONAS_16_64 + PERSONAS_65_O_MAS) %>%
  group_by(ISLA) %>%
  summarise(turistas.mas.16.total = sum(PESO),
            gasto.medio = sum(gasto.total/turistas.mas.16/NOCHES*PESO)/turistas.mas.16.total) %>%
  ungroup
```

```
## # A tibble: 7 x 3
##   ISLA                turistas.mas.16.total gasto.medio
##   <chr>                               <dbl>       <dbl>
## 1 EL HIERRO_LA GOMERA                63118.        115.
## 2 FUERTEVENTURA                    1659115.        139.
## 3 GRAN CANARIA                     3702777.        141.
## 4 LA PALMA                          235559.        131.
## 5 LANZAROTE                        2521668.        135.
## 6 No procede                         53214.        165.
## 7 TENERIFE                         5040382.        140.
```

Ejercicios propuestos

Obtener el número de turistas y el gasto medio por turista y noche en Canarias.
Obtener el número de turistas y el gasto medio por turista y noche en Canarias según el país de residencia.
Obtener el número de turistas y el gasto medio por turista y noche en cada isla según el país de residencia.

Visualización de datos con ggplot2

ggplot2

Es la librería habitual para la elaboración de gráficos en R.

Los gráficos elaborados con `ggplot2` presentan al menos tres elementos:

- **Datos**. Se recogen en el parámetro `data`.
- **Características estéticas**. Se recogen en el parámetro `mapping` mediante la función `aes()`.
- **Objetos geométricos**. Consiste en añadir una capa en la que se indique la forma de visualización. Ejecutando el comando `help.search("geom_", package = "ggplot2")` se muestra el listado de formas geométricas.

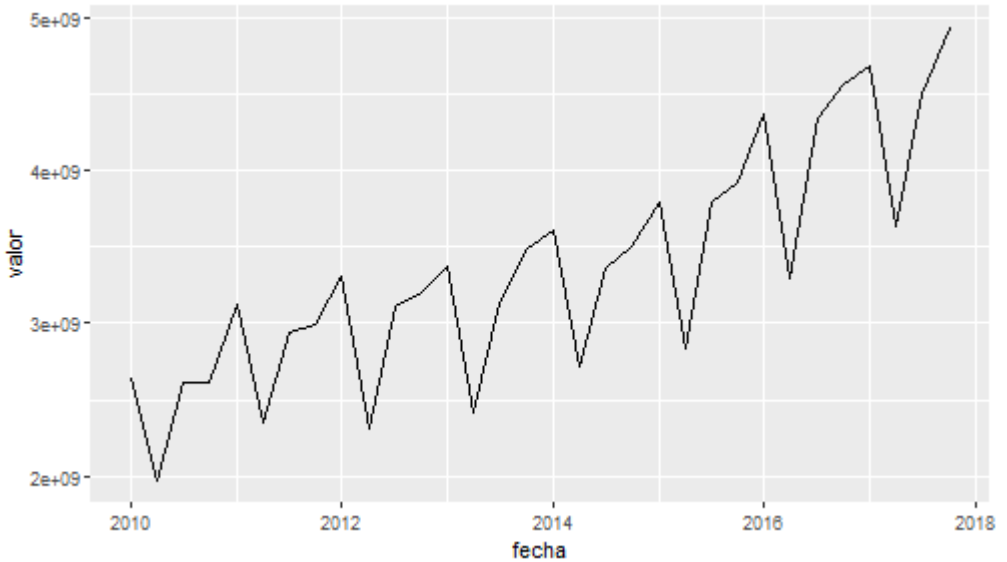
- Representar la serie temporal de gasto turístico en Canarias por trimestres.

Descargar los datos con el paquete `istacbaser`.

```
# devtools::install_github("ropenspain/istacbaser")
library(istacbaser)
busqueda.egt <- istacbase_search("egt", fields = "datos publicadosII")
df <- istacbase(busqueda.egt$ID[1], POSIXct = TRUE, freq = "trimestral")
df.trimestral <- filter(df,
                        Indicadores == "Valor absoluto" &
                        `Indicadores de gasto` == "GASTO TOTAL" &
                        `Países de residencia` == "TOTAL PAÍSES")
```

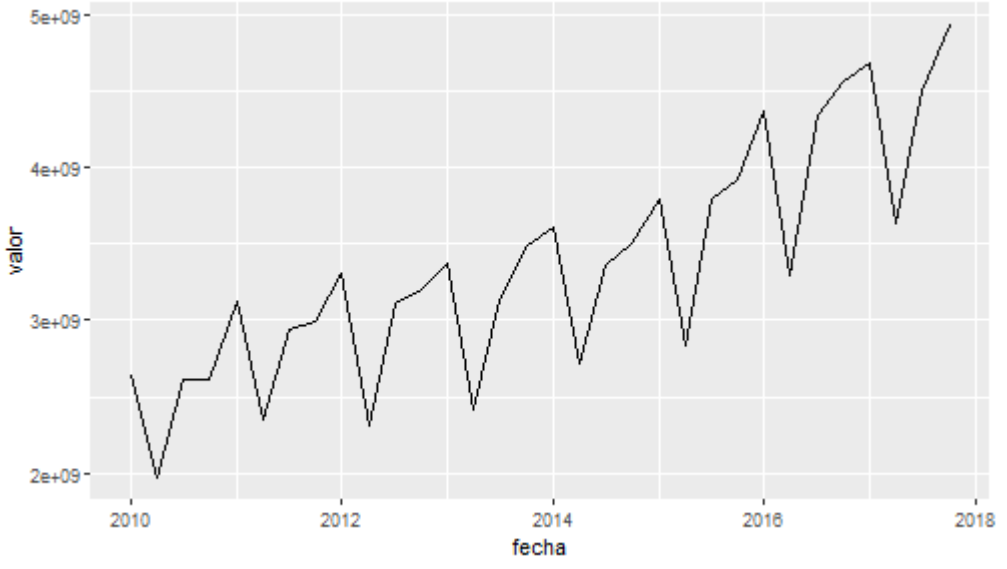
Crear el gráfico de líneas considerando al eje `x` los trimestres y al eje `y` el valor de gasto.

```
ggplot(data = df.trimestral, mapping = aes(x = fecha, y = valor))  +
  geom_line()
```



Indicar el color y el tipo de línea.

```
ggplot(data = df.trimestral, mapping = aes(x = fecha, y = valor))  +
  geom_line(
  color = "#000000",
  linetype = "solid"
  )
```

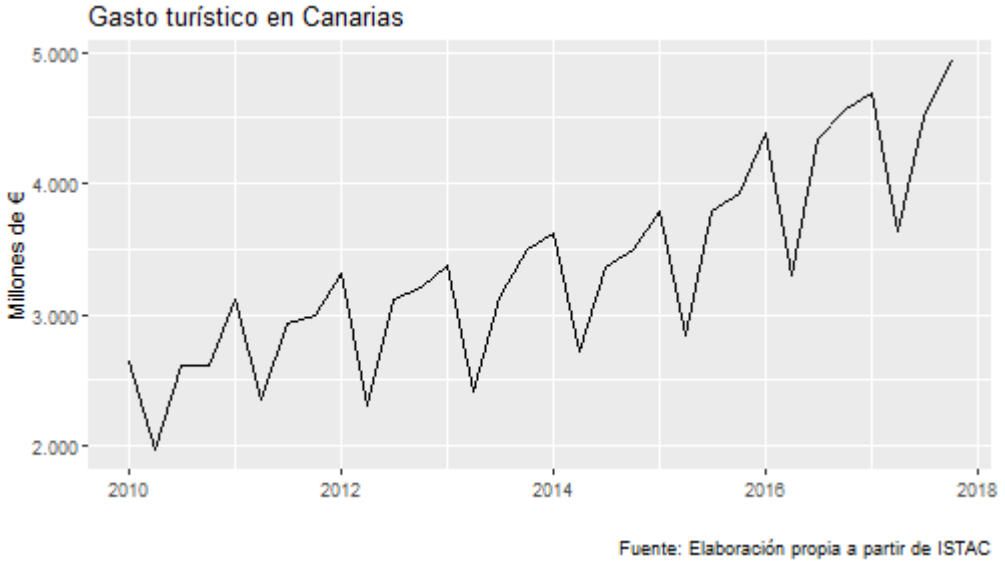


Ejercicios

Ajuste el color y el tipo de línea. Incluya una capa con forma geométrica de puntos aplicando `geom_point()`. Ajuste el color y el tamaño los puntos mediante los parámetros `color` y `size`.

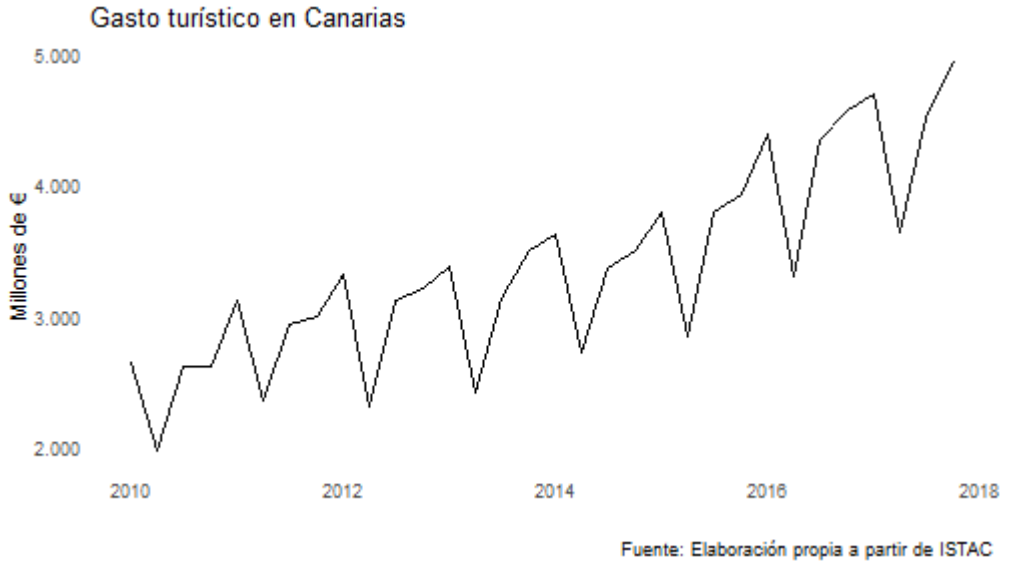
Modificar los ejes.

```
ggplot(data = df.trimestral %>% mutate(valor = valor / 1000000),
       mapping = aes(x = fecha, y = valor))  +
  geom_line(color = "#000000", linetype = "solid") +
  scale_y_continuous(labels = function(x) format(x, big.mark = ".", decimal.mark = ",",
                                                 scientific = FALSE)) +
  labs(
    title = "Gasto turístico en Canarias",
    x = "",
    y = "Millones de €",
    caption = "Fuente: Elaboración propia a partir de ISTAC"
  )
```



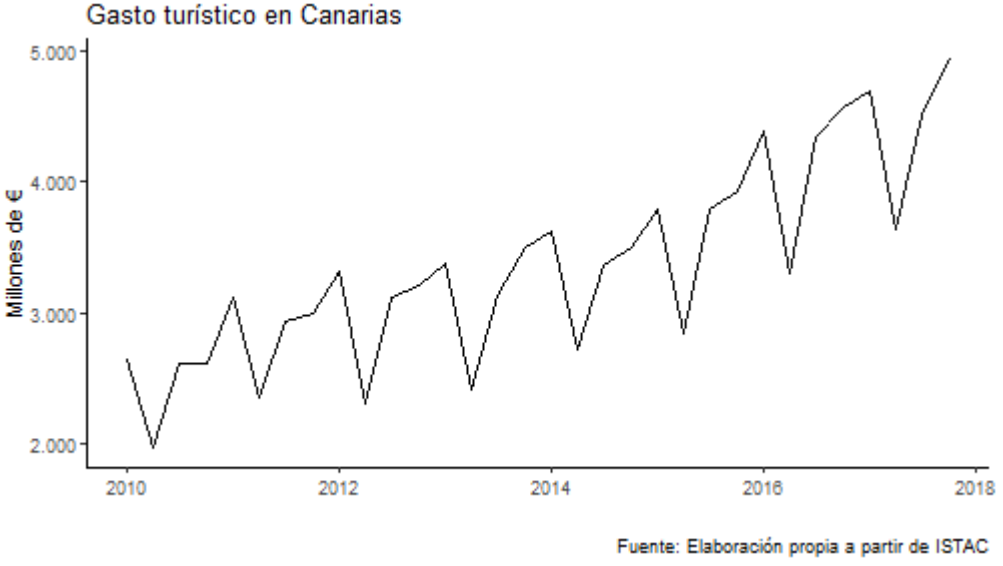
Ajustar el `theme()`.

```
ggplot(data = df.trimestral %>% mutate(valor = valor / 1000000),
       mapping = aes(x = fecha, y = valor))  +
  geom_line(color = "#000000", linetype = "solid") +
  theme(
    panel.background = element_blank(),
    axis.ticks = element_blank(),
  ) +
  scale_y_continuous(labels = function(x) format(x, big.mark = ".", decimal.mark = ",",
                                                 scientific = FALSE)) +
  labs(
    title = "Gasto turístico en Canarias",
    x = "",
    y = "Millones de €",
    caption = "Fuente: Elaboración propia a partir de ISTAC"
  )
```



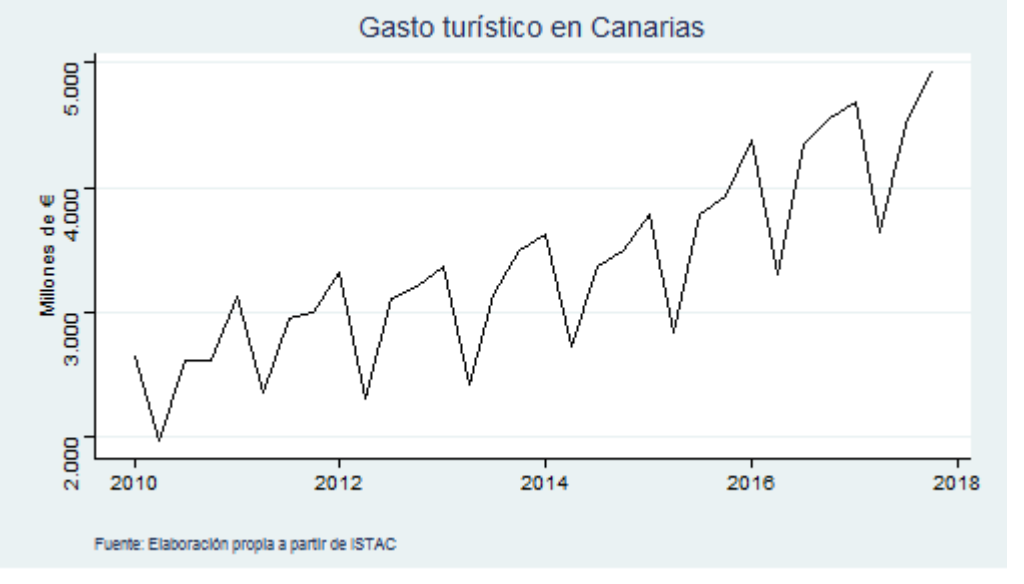
Una alternativa es emplear alguna configuración establecida.

```
ggplot(data = df.trimestral %>% mutate(valor = valor / 1000000),
       mapping = aes(x = fecha, y = valor))  +
  geom_line(color = "#000000", linetype = "solid") +
  theme_classic() +
  scale_y_continuous(labels = function(x) format(x, big.mark = ".", decimal.mark = ",",
                                                 scientific = FALSE)) +
  labs(
    title = "Gasto turístico en Canarias",
    x = "",
    y = "Millones de €",
    caption = "Fuente: Elaboración propia a partir de ISTAC"
  )
```



La librería `ggthemes` tiene múltiples opciones, por ejemplo la configuración de STATA.

```
library(ggthemes)
ggplot(data = df.trimestral %>% mutate(valor = valor / 1000000),
       mapping = aes(x = fecha, y = valor))  +
  geom_line(color = "#000000", linetype = "solid") +
  theme_stata() +
  scale_y_continuous(labels = function(x) format(x, big.mark = ".", decimal.mark = ",",
                                                 scientific = FALSE)) +
  labs(
    title = "Gasto turístico en Canarias",
    x = "",
    y = "Millones de €",
    caption = "Fuente: Elaboración propia a partir de ISTAC"
  )
```



Ejercicios

Ajuste la configuración a `theme_economist()`.

Informes con RMarkdown

RMarkdown

RMarkdown es un formato que permite elaborar documentos en R.

La principal ventaja de los documentos RMarkdown (`.Rmd`) frente a otras alternativas, como Microsoft WORD, es que permite combinar el texto con el análisis realizado en R.

Es de bastante utilidad cuando elaboramos informes recurrentes para reportar resultados, pues al cambiar los datos el informe se actualizará automáticamente.

Para crear un documento `.RMD` vamos a:

- Ir a la pestaña `File`.
- Hacer click en `New File`.
- Hacer click en `R Markdown...`

Los documentos RMarkdown tienen tres partes:

- Encabezado.
- Trozos de código R (chunks).
- Texto.

Ejercicio

Realice los ejercicios propuestos en el fichero `Ejercicio de RMarkdown.Rmd`.

Recursos para continuar con su aprendizaje

Ampliar conocimientos

Bibliografía del curso:

- Curso de introducción a R
- Análisis de datos con R en la investigación en Turismo, Economía y Gestión

Fuentes de información con las que continuar aprendiendo a trabajar con R.

- R
- RStudio
- Rbloggers
- The R Graph Gallery
- RMarkdown
- Shiny

Grupos de usuarios de R recomendados.

- RHispano
- RCanarias

¡Gracias por su atención!