

Introducción al lenguaje R

para la investigación en Ciencias de la Salud

Jaime Pinilla - C. González-Martel
03/10/2022

Introducción a R

1. Instalación y primeros pasos con R.
2. Tipos de objetos: vectores, matrices, data frames y listas.
3. Data Frames: Manejo de ficheros o conjuntos de datos.
4. Creación de funciones.
5. Instalación y trabajo con paquetes.

Instalación y primeros pasos con R.



¿Qué es R?

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

- R es un ambiente de programación formado por un conjunto de herramientas muy flexibles que pueden ampliarse fácilmente mediante paquetes, librerías o mediante funciones creadas por cualquier usuario.
- Es gratuito y de código abierto.
- R software al ser de código abierto no tiene restricciones en la modificación del código exceptos las propios de la licencia GPL, al contrario de lo que sucede con otras herramientas estadísticas comerciales privativas como SPSS, Matlab, etc.
- Es un lenguaje orientado a objetos.
- Es un lenguaje interpretado.



RStudio

RStudio

RStudio es un entorno de desarrollo integral (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)



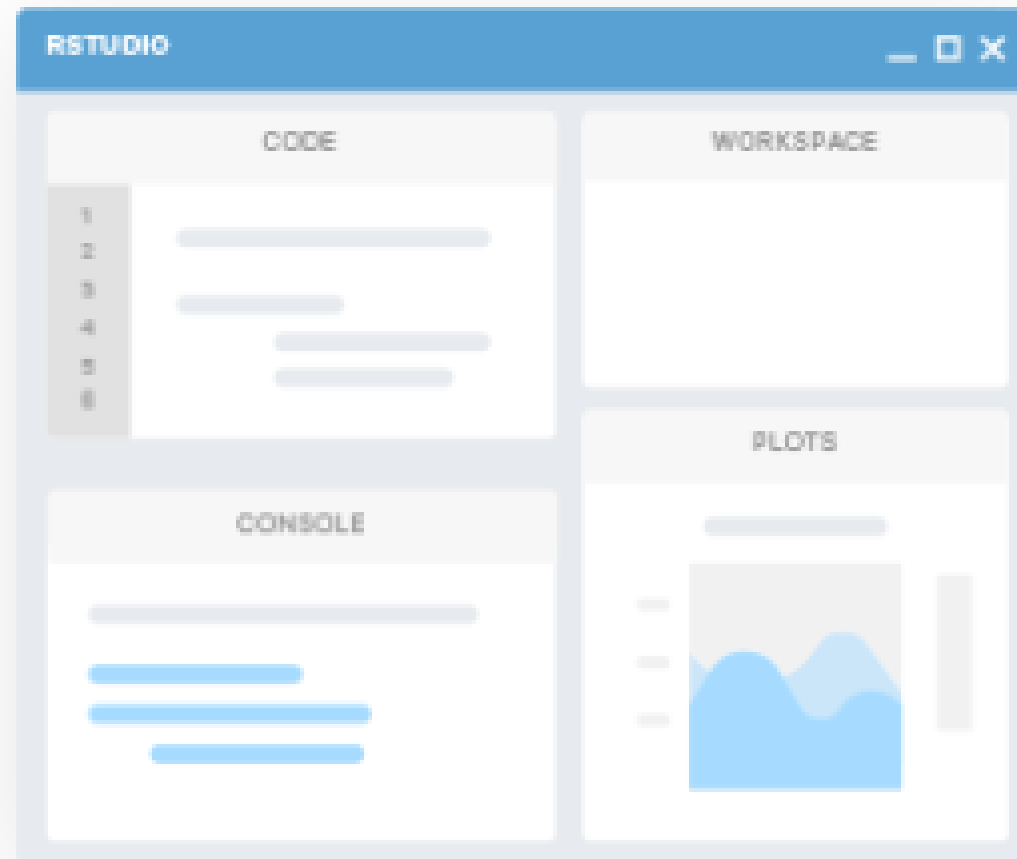
Instalación

R

- [Web de R](#)
- [Link de descarga.](#)

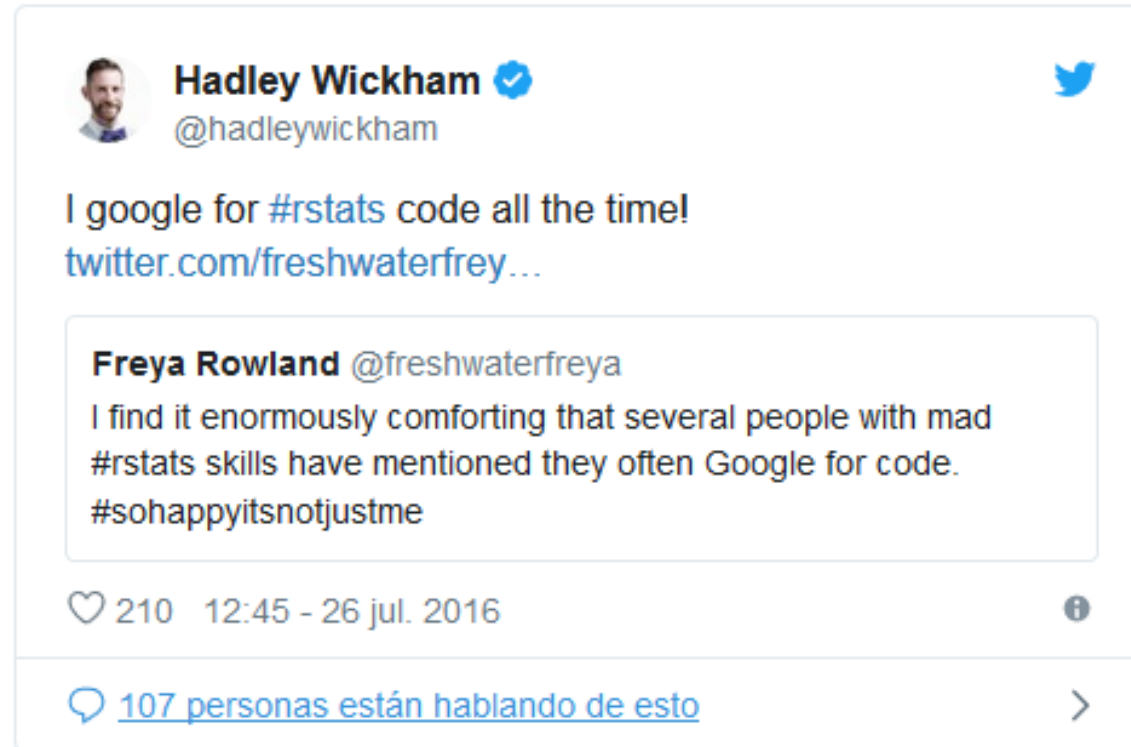
RStudio

- [Web de RStudio](#)
- [Link de descarga.](#)



I google for [#rstats](#) code all the time! <https://t.co/XDzuvqkJCK>

— Hadley Wickham (@hadleywickham) 26 de julio de 2016





Tipos de objetos: vectores, listas, matrices, y data frames.

Ejecución de nuestro primer código

En su forma más básica R funciona como una calculadora.

Ejemplo

En la consola introducir el siguiente código:

```
3 + 4
```

Ejercicio

- Calcular $12 * 3$

Operadores aritméticos

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

Ejercicio

- De nuevo usando la consola, calcular si 13 es par o impar mediante el operador módulo %%

Variables

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

Una variable permite almacenar un valor o un objeto en R. Se puede acceder al valor o al objeto almacenado en esa variable mediante el nombre de la variable.

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

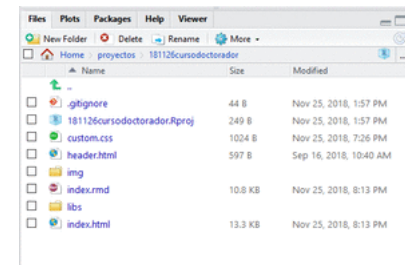
```
x
```

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

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


Establecer la carpeta de trabajo

- Vía menú
 - Ir a la pestaña Files
 - Hacer click en ... (Go to directory)
 - Seleccionar directorio
 - Hacer click en la subpestaña 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.

Editor de R

- Se puede escribir el código en un editor de texto (cualquiera) para tenerlo guardado en caso de querer ejecutarlo en futuras sesiones.
- RStudio tiene incorporado un editor con multiples funcionalidades:
 - **Manejo de ficheros**. Creación de un fichero script con File, New File
 - **Autocompletado de código**
 - **Búsqueda y reemplazo**
 - **Comentar en el código** con #
 - **Ejecución de código** con Ctrl+Enter

Ejemplo

Asignar el valor 4 a la variable `x`. Asignar el valor 5 a la variable `y`. Sumar ambos valores y asignar el resultado a la variable `suma`. Comprobar el resultado visualizando el valor de `suma`

```
# Asignar el valor '4' a la variable 'x'. Asignar el valor '5' a la variable 'y'.  
# Sumar ambos valores y asignar el resultado a la variable 'suma'.  
# Comprobar el resultado visualizando el valor de 'suma'
```

```
x <- 4  
y <- 5  
suma <- x + y  
suma
```

Escribir el código en el editor (o copiarlo desde este documento), seleccionar el código y ejecutarlo con `Ctrl+Enter` (también se puede ejecutar línea por línea). El resultado de la ejecución del código se podrá ir viendo por la consola a medida que se ejecuta.

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

Se puede guardar el código anterior a través del menú `File, Save`. Se puede recuperar en otra sesión con `File, Open`

Tipos de datos básicos en R

- Numérico num: 3.4
- Entero int: 3L
- Lógico logi: TRUE
- Texto (*string*) chr: 'char '

Con la función `class(x)` se puede obtener el tipo de dato de la variable u objeto `x`

Ejemplo

Asignar a las variables numerico, logico, texto los valores 3.4, TRUE, curso. Mediante la función class(x) obtener el tipo de las variables anteriores. Finalmente sumar numerico más texto (sin asignárselo a ninguna variable) y sumar numerico más logico

```
#Asignar a las variables 'numerico', 'logico', 'texto' los valores '3.4', 'FALSE', 'curso'. Mediante la función 'class'  
numerico <- 3.4  
logico <- TRUE  
texto <- "curso"  
numerico + texto
```

```
## Error in numerico + texto: argumento no-numérico para operador binario
```

```
numerico + logico
```

```
## [1] 4.4
```

Vectores

Los vectores son *arrays* unidimensionales. Es un agrupamiento de valores de un mismo tipo. Se declaran con la función `c()`

Ejemplo

Asignar la serie de valores 1,2,3; TRUE, FALSE, FALSE y "curso", "de", "R" a las variables `vnum`, `vlog`, `vchar`. Obtener sus clases.

```
# Asignar la serie de valores `1,2,3`, `TRUE, FALSE, FALSE` y `"curso", "de", "R"` a las variables `vnum`, `vlog`,  
vnum <- c(1,2,3)  
vlog <- c(TRUE, FALSE, FALSE)  
vchar <- c("curso", "de", "R")
```

```
class(vnum)
```

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

```
class(vlog)
```

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

```
class(vchar)
```

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

Ejercicio

Asignar la serie de valores 1,2,"texto"; 1, 2, TRUE y "curso", "de", TRUE a las variables vec1, vec2, vec3. Obtener el tipo de cada vector.

Operaciones con vectores numéricos (no exhaustiva)

Ejemplos

- Declarar el vector `vec1` con la serie 1, 3, -5, 3, -2 y el vector `vec2` con la serie 1:5

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

- Crear un nuevo vector `uvec` que sea la combinación del vector `vec1` y `vec2` y mostrar el resultado

```
uvec <- c(vec1, vec2)
uvec
```

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

- Sumar 4 a cada elemento del vector, asignárselo a la variable `suma4` y obtener el vector resultado de la operación.

```
suma4 <- vec1 + 4  
suma4
```

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

- Calcular el vector resultado de multiplicar cada elemento del vector `vec1` al elemento del vector `vec2` que ocupa la misma posición, asignárselo a la variable `prod2` para posteriormente mostrar el resultado.

```
prod2 <- vec1 * vec2  
prod2
```

```
## [1] 1 6 -15 12 10
```

- Calcular la suma de los elementos del vector `vec1`.

```
sum(vec1)
```

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

- Calcular el tamaño del vector `vec2`

```
length(vec2)
```

```
## [1] 5
```

- Calcular si los valores de `vec1` son mayores que los valores de `vec2` que ocupan la misma posición, asignarle el resultado a la variable `logico` y mostrar el resultado.

```
logico <- vec1 > vec2  
logico
```

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

Comparadores

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

Operaciones con vectores de texto. (no exhaustiva)

- Crear el vector de texto `texto1` con los valores `'curso'`, `'de'`, `'R'` y concatenar los valores de `texto1` con los de `vec1`.

```
texto1 <- c('curso', 'de', 'R')  
paste(texto1, vec1)
```

```
## [1] "curso 1" "de 3"    "R -5"    "curso 3" "de 2"
```

- Concatenar los elementos del vector `texto1` para formar un único *string*.

```
paste(texto1, collapse = " ")
```

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

Selección de elementos de vectores

Esta operación se realiza encerrando entre corchetes la posición del elemento que queremos extraer si es único, o un vector de posiciones o un vector lógico si son varios. También se pueden nombrar los elementos del vector y seleccionar en función de su nombre.

Ejemplos

- Sea el vector `vec1 <- c(1,2,-2,-5,7,12,9)` extraer el segundo elemento.

```
vec1 <- c(1,2,-2,-5,7,12,9)
vec1[2]
```

```
## [1] 2
```

- Extraer el elemento 2 y 4.

```
vec1[c(2,4)]
```

```
## [1] 2 -5
```

- Nombrar los elementos del vector `vec1` como "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo". Seleccionar el elemento correspondiente al miercoles.

```
names(vec1) <- c("lunes", "martes", "miercoles", "jueves",  
                "viernes", "sabado", "domingo")  
vec1[c("miercoles")]
```

```
## miercoles  
##      -2
```

- Extraer los elementos pares del vector `vec1`.

```
index <- vec1 %% 2 == 0  
vec1[index]
```

```
##      martes miercoles      sabado  
##          2         -2          12
```

Ejercicio

- Extraer los elementos impares del vector `vec1`.

Se pueden usar valores negativos si lo que queremos hacer es seleccionar pero eliminando elementos del vector.

- Seleccionar todos los elementos excepto el primer elemento del vector `vec2`.

```
vec2[-1]
```

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

Las selecciones de los primeros elementos y últimos se hace con las funciones `head()` y `tail()`

- Seleccionar los primeros 5 elementos del vector `vec2`.

```
head(vec2)
```

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

- Seleccionar los 3 últimos elementos del vector `vec2`.

```
tail(vec2, 3)
```

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

Asignar nuevos valores a elementos seleccionados

Ejemplos

- Asignar al segundo elemento del vector `vec2` el valor `-2` y mostrar los valores del vector `vec2`.

```
vec2[2] <- -2  
vec2
```

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

Ejercicio

- Cambiar los valores de `sabado` y `domingo` del vector `vec1` por los valores `6` y `7`, respectivamente.

Matrices

Una matriz es un objeto bidimensional del mismo tipo de datos (numéricos, enteros, lógicos o texto) repartidos en filas y columnas

- Se crean principalmente a través de la función `matrix()`

Ejemplo

Crear la matriz $\begin{bmatrix} 1, 3, 5 \\ 7, 9, 11 \end{bmatrix}$, asignársela a la variable `matriz1` y mostrar el resultado y el tipo de datos que maneja dicha variable.

```
matriz1 <- matrix(seq(1,11,2), nrow = 2, ncol = 3, byrow = TRUE)
matriz1
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    7    9   11
```

```
class(matriz1)
```

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

Otra opción hubiese sido manejando las filas o columnas como vectores. Por ejemplo, creando dos vectores como las filas de la matriz

```
vec1 <- seq(1,5,2)
vec2 <- seq(7,11,2)
matriz1 <- matrix(c(vec1,vec2), nrow = 2, byrow = T)
matriz1
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    7    9   11
```

Se puede nombrar las filas y columnas de la matriz mediante las funciones `rownames()` y `colnames()`

Ejercicio

Crear una nueva matriz `matriz2` que sea igual a `matriz1` y nombrar las filas como "fila1", "fila2" y las columnas como "columna1", "columna2", "columna2"

Selección de elementos de una matriz

En este caso hay que seleccionar la posición de la fila y de la columna separados por una coma si es un elemento individual o un vector de posiciones, lógico o de nombres para las filas y columnas separados por comas.

Ejemplos

- Seleccionar el elemento que ocupa la posición 2 en la fila y 3 en la columna de la `matriz1`.

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

```
## [1] 11
```

- Seleccionar la primera fila

```
matriz1[1, c(1,2,3)] # matriz[1,]
```

```
## [1] 1 3 5
```

Se pueden realizar operaciones por filas y columnas como por ejemplo `rowSums()`, `colSums()`, `rowMeans()`, `colmeans()`. Para realizar otro tipo de operaciones por filas o columnas se necesita la función `apply()`.

Ejemplos

- Calcular la suma de los elementos de cada columna de la matriz `matriz1`.

```
colSums(matriz1)
```

```
## [1]  8 12 16
```

- Calcular el producto de los elementos de cada fila de la matriz `matriz1`.

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

```
## [1] 15 693
```

Factores

Un factor es la forma que tiene R de almacenar variables categóricas

La forma de declarar un factor es con la función `factor()`.

Ejemplo

Declarar como factor el vector `c('Hombre', 'Mujer', 'Mujer', 'Mujer', 'Hombre')` que recoge el sexo de 5 encuestados y guardar el resultado en la variable `sexo`. Mostrar el contenido de `sexo`.

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

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

En otras ocasiones nos interesa trabajar con valores ordenados.

Ejemplo

Declarar como factor el vector `c('primarios', 'secundarios', 'superiores', 'superiores', 'secundarios')` que recoge el nivel de estudio de 5 encuestados y guardar el resultado en la variable `estudios`. Mostrar el contenido de `estudios`.

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

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

En estos casos se pueden comparar los elementos de dicho factor. Sin embargo esta compareación no es posible en variables nominales


```
estudios[1] > estudios[3]
```

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

Data Frame

Es un objeto bidimensional que contiene diferentes tipos de datos. Los datos en cada columna sí deben ser del mismo tipo.

Se declaran mediante la función `data.frame()`.

Ejemplo

Crear un *dataframe* encuesta que muestre el resultado de una encuesta a 5 personas. Se le preguntaron sobre cuestiones. Sus respuestas fueron `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")` y mostrar el contenido de la variable.

```
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(sexo, ingresos, residente, isla,
                      stringsAsFactors = FALSE)
encuesta
```

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

Se puede obtener la estructura interna del *dataframe* (en realidad de cualquier objeto de R) con la función `str()`.

Ejemplo

Obtener la estructura de la variable encuesta

```
str(encuesta)
```

```
## 'data.frame':    5 obs. of  4 variables:
## $ sexo      : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 2 1
## $ ingresos : num  1500 2300 1700 900 2100
## $ residente: logi  TRUE TRUE TRUE FALSE TRUE
## $ isla     : chr  "Gran Canaria" "Tenerife" "Tenerife" NA ...
```

Selección de elementos de un *dataframe*

Al igual que con vectores y matrices la selección de elementos del *dataframe* se realiza con los operadores `[]`. Para seleccionar los primeros y últimos datos de la tabla se usan las funciones `head()` y `tail()`

Ejercicios

- Seleccionar aquellos encuestados con ingresos superiores a los 1500 (con todas sus columnas).
- Seleccionar los ingresos e isla del primer, segundo y quinto encuestado.

El operador \$.

Este operador permite seleccionar columnas enteras de la tabla.

Ejemplo

Seleccionar la columna de isla de la tabla y asignársela a la variable `cisla`.

```
cisla <- encuesta$isla  
cisla
```

```
## [1] "Gran Canaria" "Tenerife"      "Tenerife"      NA                "Gran Canaria"
```

En muchas ocasiones no se necesita seleccionar las observaciones que cumplan cierta condición, sino que interesa conocer qué posición de la tabla ocupan esos registros. La función `which()` puede ayudarnos a detectar esas observaciones.

Ejemplo

Determinar los registros de encuestas que superen los 1500.

```
which(encuesta$ingresos > 1500)
```

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

Combinar dos *dataframes*

Se realiza con las funciones `rbind()` y `cbind()`.

Ejemplo

Sea el *dataframe* `dfc <- data.frame(hijos = round(rnorm(5,2,1),0), marca = c("mercedes", "bmw", "audi", NA, "vw"), stringsAsFactors = FALSE)`. Crear un *dataframes* `cencuesta` que sea la combinación por columnas de encuesta y `dfc`. Mostrar la estructura de `cencuesta`.

```
set.seed(123)
dfc <- data.frame(hijos = round(rnorm(5,2,1),0),
                  marca = c("mercedes", "bmw", "audi", NA, "vw"),
                  stringsAsFactors = FALSE)
cencuesta <- cbind(encuesta, dfc)
str(cencuesta)
```



```
## 'data.frame':    5 obs. of  6 variables:
## $ sexo           : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 2 1
## $ ingresos       : num  1500 2300 1700 900 2100
## $ residente      : logi  TRUE TRUE TRUE FALSE TRUE
## $ isla           : chr   "Gran Canaria" "Tenerife" "Tenerife" NA ...
## $ hijos          : num   1 2 4 2 2
## $ marca....c...mercedes....bmw....audi...NA...vw...: chr   "mercedes" "bmw" "audi" NA ...
```

Sea el *dataframe* `dfr <- data.frame(sexo = c("Hombre", "Mujer"), ingresos = c(1250, 3000), residente = c(F, T), isla = c(NA, "Fuerteventura"), stringsAsFactors = FALSE)`. Crear un *dataframes* `rencuesta` que sea la combinación por filas de encuesta y `dfr`

```
dfr <- data.frame(sexo = c("Hombre", "Mujer"),
                  ingresos = c(1250, 3000),
                  residente = c(F, T),
                  isla = c(NA, "Fuerteventura"),
                  stringsAsFactors = FALSE)
rencuesta <- rbind(encuesta, dfr)
str(rencuesta)
```

```
## 'data.frame':    7 obs. of  4 variables:
## $ sexo      : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 2 1 1 2
## $ ingresos  : num  1500 2300 1700 900 2100 1250 3000
## $ residente: logi  TRUE TRUE TRUE FALSE TRUE FALSE ...
## $ isla      : chr  "Gran Canaria" "Tenerife" "Tenerife" NA ...
```

Lista

Una lista es un objeto que permite recoger objetos de diferentes tipos

Se crean con la función `list()`

Ejemplo

Crear una lista con nombre `lista1` con los vectores `vec1`, `vec2`, la matriz `matriz1` y el *data.frame* `encuesta`.
Mostrar la estructura de la lista recién creada.

```
lista1 <- list(vec1, vec2, matriz1, encuesta)
str(lista1)
```

```
## List of 4
## $ : num [1:3] 1 3 5
## $ : num [1:3] 7 9 11
## $ : num [1:2, 1:3] 1 7 3 9 5 11
## $ : 'data.frame': 5 obs. of 4 variables:
## ..$ sexo : Factor w/ 2 levels "Hombre","Mujer": 1 2 2 2 1
## ..$ ingresos : num [1:5] 1500 2300 1700 900 2100
## ..$ residente: logi [1:5] TRUE TRUE TRUE FALSE TRUE
## ..$ isla : chr [1:5] "Gran Canaria" "Tenerife" "Tenerife" NA ...
```

Seleccionar elementos de una lista

Los elementos de una lista se seleccionan con el operador `[[]]`, o bien con el operador `$` si el elemento dentro de la lista tiene nombre. Una vez seleccionado, a su vez, se pueden seleccionar elementos dentro de ese elemento con el operador `[]`.

Ejemplo

- Seleccionar el segundo elemento del vector `vec1` de la lista `lista1`

```
lista1[[1]][2]
```

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

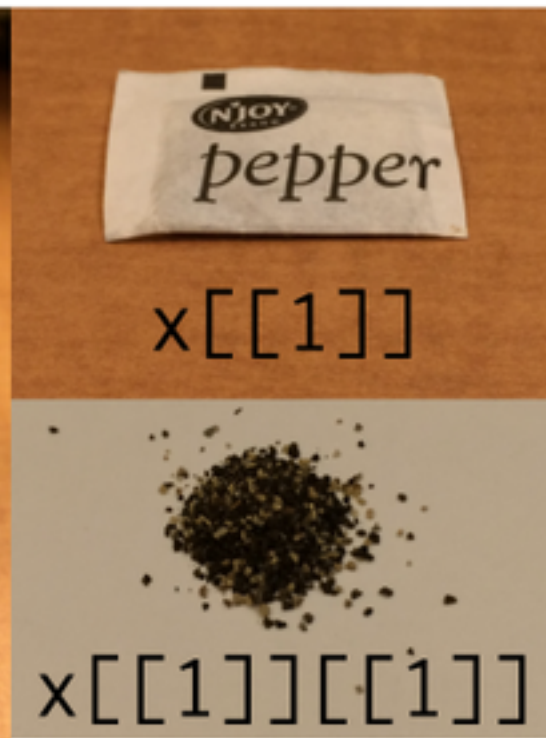
```
# lista$vec1[2]
```



x



x[1]



x[[1]]

x[[1]][[1]]



Data Frames: Manejo de ficheros o conjuntos de datos.

- La lectura de los datos es una etapa fundamental antes de empezar con la exploración de los datos y el posterior análisis.
- La mayoría de las veces esa lectura o importación se hacen desde ficheros o fuentes externas.
- Es el primer paso del *workflow* para el análisis de datos.
 - Lectura o importación de datos
 - Limpieza
 - Exploración
 - Análisis

Importar ficheros csv

- Con el comando `read.csv()`¹

Ejemplo

Leer el fichero `gasto_clean.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`. Cambiar el nombre de las variables por `c("Pais", paste0("T", 1:3, "_2018"))` Mostrar la estructura y las 5 primeras filas.

```
datos <- read.csv2("data/gasto_clean.csv", stringsAsFactors = FALSE)
names(datos) <- c("Pais", paste0("T", 1:3, "_2018"))
str(datos)
head(datos)
```

[1] Considerar la función `read.csv2()` cuando el separador es ';'. Parámetros a considerar `skip` y `header`

```
## 'data.frame':  6 obs. of  4 variables:
## $ Pais      : chr  "Alemania" "España" "Holanda" "Países Nórdicos" ...
## $ T1_2018: chr  "2.149,02" "1.789,30" "2.206,18" "1.875,08" ...
## $ T2_2018: chr  "2.281,34" "1.563,97" "1.832,84" "2.237,31" ...
## $ T3_2018: chr  "2.122,84" "1.279,85" "1.595,82" "2.175,26" ...
```

```
##      Pais T1_2018 T2_2018 T3_2018
## 1  Alemania 2.149,02 2.281,34 2.122,84
## 2  España  1.789,30 1.563,97 1.279,85
## 3  Holanda 2.206,18 1.832,84 1.595,82
## 4 Países Nórdicos 1.875,08 2.237,31 2.175,26
## 5  Reino Unido 2.579,05 1.813,21 1.641,40
## 6  Otros países 2.730,37 1.803,99 2.065,13
```

Para terminar de limpiar los datos debemos convertir los textos en números. Normalmente se podría hacer a través de `as.numeric`. Sin embargo, al estar escritos los números en formato no anglosajón hay que hacer la conversión de forma manual. Hay que eliminar el punto, sustituir la , por el . y convertirlo con `as.numeric`.

```
datos$T1_2018 <- gsub("\\\\.", "", datos$T1_2018)
datos$T1_2018 <- gsub(",", "\\.", datos$T1_2018)
datos$T1_2018 <- as.numeric(datos$T1_2018)
class(datos$T1_2018)
```

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

Ejercicio

Repetir la operación para las columnas `T2_2018` y `T3_2018` y mostrar las 5 primeras filas de la tabla.

- Mediante la función `read_csv()` del paquete `readr`.²

```
install.packages("readr")
```

```
library(readr)
datos <- read_csv2("data/gasto_clean.csv")
```

```
## i Using ",",'" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
## Rows: 6 Columns: 4
```

```
## -- Column specification -----
```

```
## Delimiter: ";"
```

```
## chr (1): País
```

```
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
names(datos) <- c("País",paste0("T",1:3,"_2018"))
## str(datos)
```

[2] Considerar el parámetro `locale = locale(encoding = 'ISO-8859-1')` cuando el csv está grabado desde excel para Windows. [Cheetsheet](#)

```
## spec_tbl_df [6 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ Pais      : chr [1:6] "Alemania" "España" "Holanda" "Países Nórdicos" ...
## $ T1_2018: num [1:6] 2149 1789 2206 1875 2579 ...
## $ T2_2018: num [1:6] 2281 1564 1833 2237 1813 ...
## $ T3_2018: num [1:6] 2123 1280 1596 2175 1641 ...
## - attr(*, "spec")=
## .. cols(
## ..   País = col_character(),
## ..   `2018 Tercer trimestre` = col_number(),
## ..   `2018 Segundo trimestre` = col_number(),
## ..   `2018 Primer trimestre` = col_number()
## .. )
## - attr(*, "problems")=<externalptr>
```

- Mediante el botón Import Dataset de la pestaña Environment

Importar ficheros excel

- Copiando y cargando los datos copiados

```
datos_excel <- read.table(file = "clipboard", sep = "\t", header=TRUE)  
str(datos_excel)
```

En este caso las columnas las ha convertido en factor. Hay que convertirlas en *strings* con la función `as.character()` y después limpiar los datos para poder convertirlos a numéricos como se hizo en ejemplos anteriores.

- Mediante la función `read_excel()` del paquete `readxl`

```
install.packages("readxl")
```

```
library(readxl)
datos_excel <- read_excel("data/gastos.xlsx")
str(datos_excel)
```

```
## tibble [6 x 4] (S3: tbl_df/tbl/data.frame)
## $ País : chr [1:6] "Alemania" "España" "Holanda" "Países Nórdicos" ...
## $ 2018 Tercer trimestre : chr [1:6] "2.149,02" "1.789,30" "2.206,18" "1.875,08" ...
## $ 2018 Segundo trimestre: chr [1:6] "2.281,34" "1.563,97" "1.832,84" "2.237,31" ...
## $ 2018 Primer trimestre : chr [1:6] "2.122,84" "1.279,85" "1.595,82" "2.175,26" ...
```

En este caso las columnas las ha convertido en *strings*. Hay que limpiar los datos para poder convertirlos a numéricos como se hizo en ejemplos anteriores.

- Mediante el botón Import Dataset de la pestaña Environment

Importar datos de otros software como Stata, SAS, SPSS

- 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	<code>save.image()</code>	<code>load()</code>
Algunos objetos	<code>save()</code>	<code>load()</code>
Un objeto	<code>saveRDS()</code> , <code>readr::write_rds()</code>	<code>readRDS()</code> , <code>readr::read_rds()</code>
Csv	<code>write.csv()</code> , <code>readr::write_csv()</code>	<code>read.csv()</code> , <code>readr::read_csv()</code>
Excel	<code>xlsx::write.xlsx()</code>	<code>readr::read_excel()</code>

Ejemplos

- Guardar todas las variable del espacio de trabajo en el fichero `cursor1asesion.Rdta` dentro de la carpeta `data`

```
save.image("data/cursor1asesion.Rdta")
```

- Añadir una nueva columna `total` a la tabla `datos` que sea la suma de los gastos de los tres trimetres y asignaremos la nueva tabla a la variable `ndatos`. Guardar los *dataframes* `datos` y `ndatos` en el fichero `datos.Rdta` en la carpeta `data`. Borrar esas dos variables y volverlas a cargar en el espacio de trabajo.

```
ndatos <- datos  
ndatos$total <- colSums(datos[, -1])  
save(datos, ndatos, "data/datos.Rdta")  
rm(datos, ndatos)  
load("data/datos.Rdta")
```

- Guardar datos en el fichero datos.Rds en la carpeta data. Cargar el objeto bajo el nombre datosrds.

```
saveRDS(datos, "data/datos.Rds")  
datosrds <- readRDS("data/datos.Rds")
```

- Guardar ndatos en el fichero ndatos.csv dentro de la carpeta data.

```
write_csv(ndatos, "data/ndatos.csv")
```

- Guardar datos y encuestas en dos hojas del fichero dataframes.xlsx dentro de la carpeta data

```
install.packages("xlsx")  
library(xlsx)  
write.xlsx(datos, file = "data/dataframes.xlsx")  
write.xlsx(encuestas, file = "data/dataframes.xlsx", append=TRUE)
```

Creación de funciones.

Condicionales

If

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

```
| if(cond){ expresión }
```

Ejemplo

Comparamos el total de gasto del primer y segundo semestre de 2018 según datos recogidos en la tabla datos y en caso de que el total del primer trimestre sea mayor que el del segundo imprimir por pantalla El total de gasto del primer trimestre de 2018 fue superior al del segundo.

```
total1T <- sum(datos$T1_2018)
total2T <- sum(datos$T2_2018)
texto <- paste0()
if(total1T > total2T){
  print("El total de gasto del primer trimestre de 2018 fue superior al del segundo")
}
```

```
## [1] "El total de gasto del primer trimestre de 2018 fue superior al del segundo"
```

Else

Mediante el uso de else se puede establece qué parte del código se ejecuta en caso de que la consición no se cumpla.

Ejemplo

Siguiendo el ejemplo anterior, imprimir el mensaje El total de gasto del segundo trimestre de 2018 fue superior al del primero en caso de que no se cumpla la condición.

```
if(total1T > total2T){  
    print("El total de gasto del primer trimestre de 2018 fue superior al del segundo")  
} else {  
    print("El total de gasto del segundo trimestre de 2018 fue superior al del primero")  
}
```

```
## [1] "El total de gasto del primer trimestre de 2018 fue superior al del segundo"
```

En la transparencia #27 se introdujeron los operadores lógicos. A esta lista se deben añadir el operador lógico **y** (&) y el operador lógico **o** (|) que controlan si se cumplen varias condiciones.

Ejemplos

- Seleccionar de la tabla `total` aquellos países que tengan en el primer trimestre de 2018 un gasto superior a 1500 e inferior a 2000.

```
index <- datos$T1_2018 > 1500 & datos$T1_2018 < 2000
datos[index,]
```

```
## # A tibble: 2 x 4
##   Pais          T1_2018 T2_2018 T3_2018
##   <chr>         <dbl>   <dbl>   <dbl>
## 1 España        1789.   1564.   1280.
## 2 Países Nórdicos 1875.   2237.   2175.
```

- Seleccionar de la tabla total aquellos países que tengan en el primer trimestre de 2018 un gasto inferior a 1500 o superior a 2000.

```
index <- datos$T1_2018 > 1500 | datos$T1_2018 < 2000
datos[index,]
```

```
## # A tibble: 6 x 4
##   Pais          T1_2018 T2_2018 T3_2018
##   <chr>         <dbl>   <dbl>   <dbl>
## 1 Alemania      2149.    2281.    2123.
## 2 España        1789.    1564.    1280.
## 3 Holanda       2206.    1833.    1596.
## 4 Países Nórdicos 1875.    2237.    2175.
## 5 Reino Unido    2579.    1813.    1641.
## 6 Otros países   2730.    1804.    2065.
```

Bucles

La estructura básica para un bucles es a través de:

```
for (var in vec){expresiones}
```

Ejemplo

Recorrer con un bucle la columna T1_2018 de la tabla datos e imprimir el valor.

```
for (i in datos$T1_2018){  
  print(i)  
}
```

```
## [1] 2149.02  
## [1] 1789.3  
## [1] 2206.18  
## [1] 1875.08  
## [1] 2579.05  
## [1] 2730.37
```

```
for (i in 1:length(datos$T1_2018)){  
  print(datos$T1_2018[i])  
}
```

```
## [1] 2149.02  
## [1] 1789.3  
## [1] 2206.18  
## [1] 1875.08  
## [1] 2579.05  
## [1] 2730.37
```

Combinando condicionales y bucles

Ejemplo

Crear un *dataframe* `datos_copia` que sea igual a `datos`. Crear en `datos_copia` la columna `gasto_mayor`, que será igual a 1 si el gasto en el primer trimestre es superior al segundo.

```
temp <- c()
for (i in 1:nrow(datos)){
  if(datos$T1_2018[i] > datos$T2_2018[i]){
    temp[i] <- 1
  } else
    temp[i] <- 0
}
datos$gasto_mayor <- temp
head(datos,3)
```

```
## # A tibble: 3 x 5
##   Pais      T1_2018 T2_2018 T3_2018 gasto_mayor
##   <chr>      <dbl>   <dbl>   <dbl>      <dbl>
## 1 Alemania   2149.    2281.    2123.         0
## 2 España     1789.    1564.    1280.         1
## 3 Holanda    2206.    1833.    1596.         1
```


Funciones

```
## spec_tbl_df [6 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ Pais      : chr [1:6] "Alemania" "España" "Holanda" "Países Nórdicos" ...
## $ T1_2018    : num [1:6] 2149 1789 2206 1875 2579 ...
## $ T2_2018    : num [1:6] 2281 1564 1833 2237 1813 ...
## $ T3_2018    : num [1:6] 2123 1280 1596 2175 1641 ...
## $ gasto_mayor: num [1:6] 0 1 1 0 1 1
## - attr(*, "spec")=
## .. cols(
## ..   País = col_character(),
## ..   `2018 Tercer trimestre` = col_number(),
## ..   `2018 Segundo trimestre` = col_number(),
## ..   `2018 Primer trimestre` = col_number()
## .. )
## - attr(*, "problems")=<externalptr>
```

Para hacer la conversión a numérico tendríamos que repetir el siguiente código para cada columna.

```
gastos$T1_2018 <- gsub("\\.", "", datos$T1_2018)
gastos$T1_2018 <- gsub(",", "\\.", datos$T1_2018)
gastos$T1_2018 <- as.numeric(datos$T1_2018)
```

Podemos crear una función y aplicarla a cada columna

```

mi_funcion <- function(columna){
  columna <- gsub("\\\\.", "", columna)
  columna <- gsub(",", "\\.", columna)
  columna <- as.numeric(columna)
  columna
}
conv <- apply(gastos[,2:4], 2, mi_funcion)
df.gastos <- data.frame(Pais = gastos$Pais, conv, stringsAsFactors = FALSE)
df.gastos

```

```

##              Pais T1_2018 T2_2018 T3_2018
## 1      Alemania 2149.02 2281.34 2122.84
## 2      España 1789.30 1563.97 1279.85
## 3      Holanda 2206.18 1832.84 1595.82
## 4 Países Nórdicos 1875.08 2237.31 2175.26
## 5      Reino Unido 2579.05 1813.21 1641.40
## 6      Otros países 2730.37 1803.99 2065.13

```

Instalación y trabajo con paquetes.

Un paquete o librería en R es una colección de funciones de R, datos y código compilado en un formato predefinido. Cuando se ejecuta R se cargan unos paquetes estándar como son los paquetes base, stats, etc. El resto se deben instalar y cargar en memoria antes de poder usar sus funciones.

Instalación de paquetes

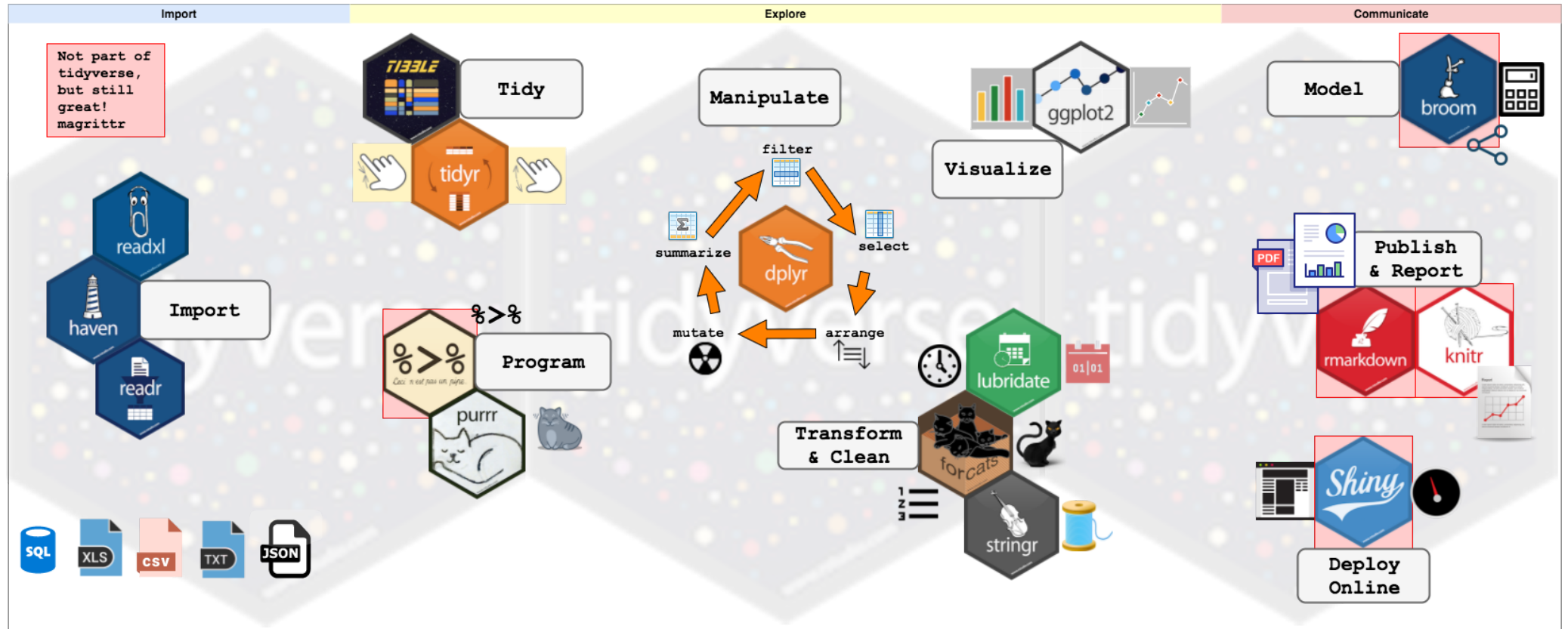
- Con el comando `install.package()`
- A través de la pestaña Packages y pulsando el botón Install
- A través del menú Tools, Install packages

Cargar el paquete en memoria

- Con el comando `library()`
- Pulsando en la casilla de verificación de la lista de paquetes instalados en la pestaña de Packages

Preparación de datos.

Tidyverse



Paquete dplyr ⁴

```
# install.package(dplyr)  
library(dplyr)
```

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.

```
# R base  
head(datos)  
# Tidyverse  
datos %>% head()
```

Funciones básicas

- `mutate`. Crea una nueva columna.
- `select`. Selecciona columnas.
- `filter`. Filtra en función de una condición.
- `group_by`. Agrupa.
- `summarise`. Resume.
- `arrange`. Ordena.

Ejemplos

- Crear el *dataframe* seleccion que contenga solo las columnas de gatos del primer y segundo trimestre de la tabla gasto

```
seleccion <- select(datos, T1_2018, T2_2018)
seleccion <- datos %>% select(T1_2018, T2_2018)
seleccion
```

```
## # A tibble: 6 x 2
##   T1_2018 T2_2018
##   <dbl>   <dbl>
## 1  2149.   2281.
## 2  1789.   1564.
## 3  2206.   1833.
## 4  1875.   2237.
## 5  2579.   1813.
## 6  2730.   1804.
```

- Crear el *dataframe* filtrado que contenga una nueva columna llamada T1_100 que sea igual a la columna T1_2018 de la tabla gasto dividida por 1000, seleccionar aquellos países cuyo valor en esta nueva columna sea mayor o igual a 2 y ordenarlos por el valor obtenido en esta columna.

```
filtrado <- datos %>%  
  mutate(T1_100 = T1_2018/1000) %>%  
  filter(T1_100 >= 2) %>%  
  arrange(T1_100)  
filtrado
```

```
## # A tibble: 4 x 6  
##   Pais          T1_2018 T2_2018 T3_2018 gasto_mayor T1_100  
##   <chr>         <dbl>  <dbl>  <dbl>      <dbl>  <dbl>  
## 1 Alemania      2149.   2281.   2123.         0    2.15  
## 2 Holanda      2206.   1833.   1596.         1    2.21  
## 3 Reino Unido   2579.   1813.   1641.         1    2.58  
## 4 Otros países  2730.   1804.   2065.         1    2.73
```

Paquete tidyr⁵

Contiene funciones para organizar datos en forma tabular.

```
# install.packages(tidyr)
library(tidyr)
```

Ejemplo

Crear la tabla `org` que contenga los valores de los tres trimestres de la tabla `datos` en una única columna.

```
org <- gastos %>%
  pivot_longer(cols = c("T1_2018", "T2_2018", "T3_2018"),
               names_to = "trimestre",
               values_to = "gasto")
org
```

```
## # A tibble: 18 x 3
```

```
##   Pais                               trimestre gasto
##   <chr>                               <chr>      <chr>
## 1 "      Alemania"                    T1_2018    2.149,02
## 2 "      Alemania"                    T2_2018    2.281,34
## 3 "      Alemania"                    T3_2018    2.122,84
## 4 "      España"                      T1_2018    1.789,30
## 5 "      España"                      T2_2018    1.563,97
## 6 "      España"                      T3_2018    1.279,85
## 7 "      Holanda"                    T1_2018    2.206,18
## 8 "      Holanda"                    T2_2018    1.832,84
## 9 "      Holanda"                    T3_2018    1.595,82
## 10 "    Países Nórdicos"              T1_2018    1.875,08
## 11 "    Países Nórdicos"              T2_2018    2.237,31
## 12 "    Países Nórdicos"              T3_2018    2.175,26
## 13 "    Reino Unido"                  T1_2018    2.579,05
## 14 "    Reino Unido"                  T2_2018    1.813,21
## 15 "    Reino Unido"                  T3_2018    1.641,40
## 16 "    Otros países"                 T1_2018    2.730,37
## 17 "    Otros países"                 T2_2018    1.803,99
## 18 "    Otros países"                 T3_2018    2.065,13
```

Ejercicio

- Importar el fichero `casos_hosp_uci_def_sexo_edad_provres.csv` desde la url https://cnecovid.isciii.es/covid19/resources/casos_hosp_uci_def_sexo_edad_provres.csv en la variable `covid19`. Recoge el número de casos, número de hospitalizaciones, número de ingresos en UCI y número de defunciones por sexo, edad y provincia de residencia de de COVID-19 declarados a a la Red Nacional de Vigilancia Epidemiológica (RENAVE) a través de la plataforma informática vía Web SiViES (Sistema de Vigilancia de España) que gestiona el Centro Nacional de Epidemiología (CNE)⁶

```
url <- "https://cnecovid.isciii.es/covid19/resources/casos_hosp_uci_def_sexo_edad_provres.csv"
covid19 <- readr::read_csv(url)
```

- Crear el *dataframe* seleccion que contenga solo las columnas provincia_iso,sexo,grupo_edad,fecha y num_casos de la tabla covid19
- Crear el *dataframe* filtrado que contenga solo los casos de la provincia de Alicante (A). A continuación crear una nueva columna llamada num_casos_rel que sea igual a la columna num_casos de la tabla covid19 dividida por 1 879 888, que fue la población de Alicante en 2020 (Fuente: [Wikipedia](#)).
- Crear el *dataframe* agrupado que sea el número total de casos para cada provincia para todo el periodo. A continuación, crear una nueva columna que represente el porcentaje de casos de cada provincia sobre el total.
- Crear el *dataframe* ancho que contenga en columnas el nombre de las provincias y en filas el número de casos por fecha. *Antes de realizar el pivot wider debemos agrupar por provincia_iso y fecha.*

Ejercicio

You see? you can do it

**I'm done
with everything
today.**