Práctica 1: comenzando con R y Rstudio

Table of Contents

Objetivos

1. Instalar R y Rstudio
2. Conocer el entorno Rstudio como herramienta básica para utilizar el programa R.
3. Identificar las distintas ventanas de Rstudio y la utilidad de cada una.
4. Conocer y comenzar a utilizar documentos Rmarkdown como soporte para integrar texto y cálculos, y para producir como resultado documentos en html o docx reproducibles y con buena presentación.
5. Aprender a instalar (en el disco) y cargar (en memoria) algunas librerías que añaden nuevas capacidades a la instalación básica de R.
6. Aprender a leer datos desde archivos Excel.
7. Comenzar a utilizar las funciones del conjunto de librerías **tidyverse** para la manipulación y análisis de datos.
8. Conocer las posibilidades del paquete **flextable** para la generación de tablas bien formateadas.

Desarrollo de la práctica: preliminares

Para el desarrollo de la práctica se podrán utilizar los ordenadores de la Facultad de Ciencias del Mar, o el ordenador portátil que haya traído cada estudiante.

Instalación

Antes de asistir a la práctica, se ha informado a los alumnos a través del campus virtual que para instalar R y Rstudio en sus ordenadores personales basta con descargar y ejecutar los programas de instalación (versiones de febrero de 2020) que encontrarán en los siguientes enlaces directos:

- **Windows**
  - R 3.6.2
  - Rstudio 1.2.5033

- **Mac**
  - R 3.6.2
  - Rstudio 1.2.5033

- **Linux**

  Los siguientes enlaces muestran como se instalan R y Rstudio dependiendo de la distribución de Linux que se utilice (debian, ubuntu, redhat, suse):
  - R 3.6.2
  - Rstudio 1.2.5033

Con los alumnos que hayan traído su ordenador portátil, durante la práctica simplemente se revisará que la instalación de R y Rstudio sea correcta o, en su caso, se ayudará a aquellos alumnos que hayan tenido alguna dificultad en la instalación.

Rmarkdown

Para la realización de las tareas que se describen a continuación se deberá crear un archivo Rmarkdown en el que se irá incluyendo el código R que se emplee, así como aquellos comentarios y texto que cada alumno juzgue oportuno para aclarar las distintas actividades que se irán realizando durante la práctica. El objetivo de utilizar este tipo de archivos es que el resultado final que se obtiene es un documento en html o word de alta calidad en el que quedan perfectamente

integrados el texto con los resultados estadísticos (incluyendo tablas y gráficos) obtenidos con R.

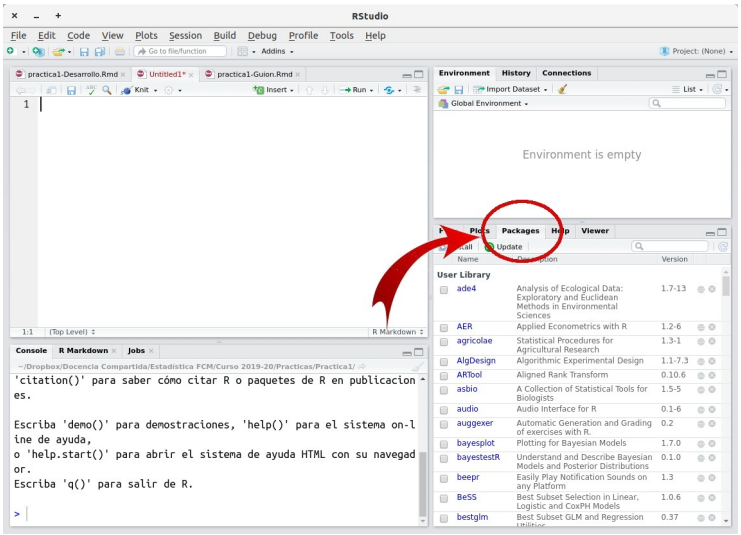
Para crear y manipular un archivo Rmarkdown una vez arrancado Rstudio basta con seguir las instrucciones de este enlace donde además se explica el concepto y manejo básico de este tipo de archivos.

También aquí o aquí se pueden encontrar excelentes introducciones (en español) al uso de Rmarkdown.

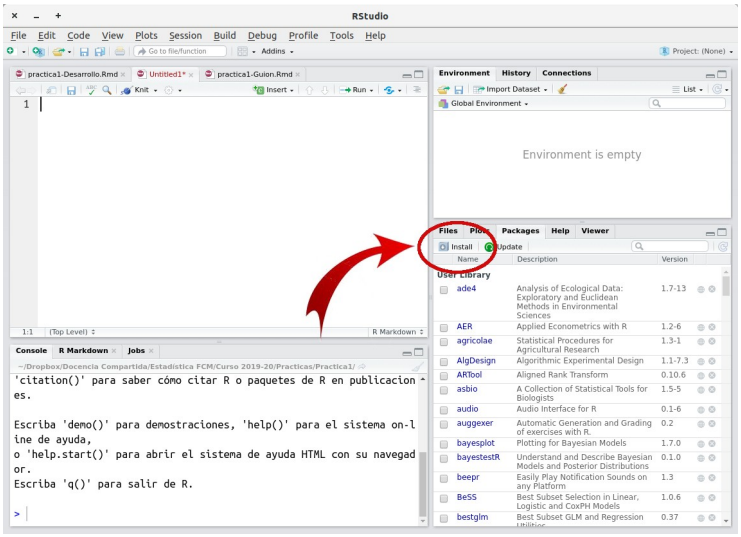
Para aquellos alumnos aficionados a los tutoriales de Youtube, la Universitat de les Illes Balears tiene tres videos dedicados a RMarkdown: 1. R markdown, 2: Chunks en Rmarkdown y 3: Figuras en Rmarkdown. Otro video que puede resultar útil es RMarkdown para principiantes

0. Instalación de librerías (Packages).

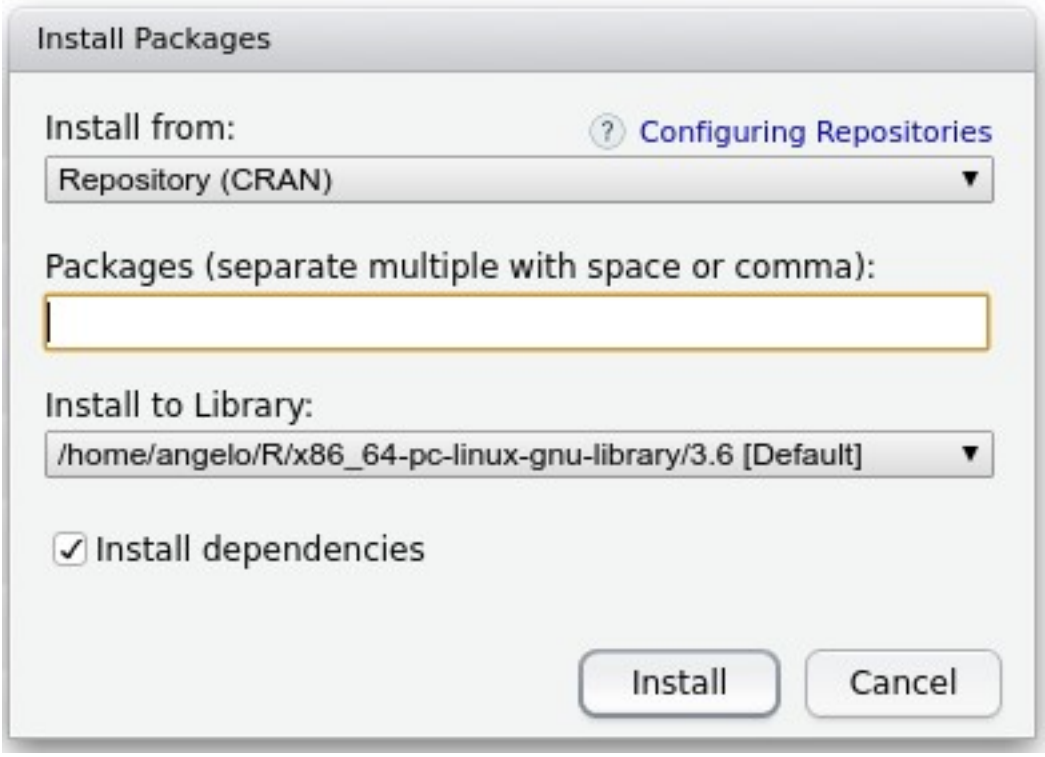
Una librería en R es una colección de funciones que amplían la capacidad de la instalación base de R. Actualmente en R hay disponibles algo más de 15.000 librerías, orientadas a todo tipo de tareas: desde la importación de datos desde casi cualquier fuente posible, a la elaboración de mapas o páginas web, pasando por la implementación de una infinidad de métodos estadísticos. Antes de poder utilizar una librería es preciso descargarla e instalarla en nuestro disco duro. Podemos hacerlo de dos formas: directamente en la consola escribiendo `install.packages("nombre del paquete a instalar")` o bien accediendo en Rstudio a la pestaña `Packages::`



y a continuación pinchando en `Install`:



Se abrirá entonces esta ventana:



y bastará escribir aquí los nombres de los paquetes que queremos instalar. Es importante que esté marcada la casilla `Install dependencies`, ya que muchos paquetes de R dependen de funciones que están contenidas en otros paquetes; es normal que cuando instalemos un paquete, R instale primero varios paquetes adicionales necesarios para que aquel funcione.

Para esta primera práctica instalaremos los siguientes paquetes:

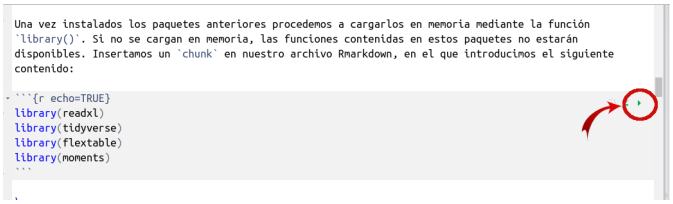
- `readxl`: para la importación de archivos Excel.
- `tidyverse`: para la organización y manipulación de datos.
- `flextable`: para la elaboración de tablas listas para presentación
- `moments`: para el cálculo de algunos estadísticos descriptivos, en particular la **asimetría** (*skewness*) y el **apuntamiento** (*kurtosis*).

1. Cargar las librerías necesarias para la manipulación y análisis de datos: `library()`

Una vez instalados los paquetes anteriores procedemos a cargarlos en memoria mediante la función `library()`. Si no se cargan en memoria, las funciones contenidas en estos paquetes no estarán disponibles. Insertamos un `chunk` en nuestro archivo Rmarkdown, en el que introducimos el siguiente contenido:

```
library(readxl)
library(tidyverse)
library(flextable)
library(moments)
```

El `chunk` quedará así:



Si pinchamos en el triangulito verde que hemos destacado a la derecha en esta imagen, se ejecutarán todas las lineas del `chunk`, y por tanto las librerías se cargarán en memoria y estaremos en condiciones de utilizarlas.

En todos los puntos que siguen en el desarrollo de esta práctica, el código R habrá de escribirse siempre dentro de un `chunk`.

2. Leer un archivo de datos en Excel: `read_excel()`

Usaremos los datos contenidos en el archivo datosP1-1920.xlsx. Estos datos son un subconjunto de los utilizados para realizar esta presentación, basada en los resultados de un muestreo de nidos de tortuga *Caretta caretta* realizado en la isla de Boavista del archipiélago de Cabo Verde. Cada fila del archivo corresponde a un nido de tortuga, y cada columna a una variable: *LCC, ACC* y *Peso* son medidas de la tortuga que hizo el nido; *Huevos* el número de huevos del nido, *distancia* es la distancia desde el nido hasta la orilla, etc.

Para leer este archivo primero lo descargamos al ordenador. Si lo guardamos en la misma carpeta en la que se encuentra el archivo Rmarkdown con el que estamos trabajando no será preciso especificar la ruta en la que se encuentra, y para leerlo bastará con ejecutar:

```
tortugas <- read_excel("datosP1-1920.xlsx")
```

Si el archivo se ha descargado, por ejemplo, en la carpeta `Descargas` deberemos averiguar la ruta exacta de esa carpeta e incluirla antes del nombre del archivo:

```
tortugas <- read_excel("c:/Users/usuario/Downloads/datosP1-1920.xlsx")
```

Téngase en cuenta que aunque en Windows la denominación de la carpeta sea `c:\Users\usuario\Downloads` (nótese que las barras van de izquierda a derecha, "\"), para que R sea capaz de leer el archivo, las barras separadoras deben escribirse de la forma "/".

Lo que hemos hecho al utilizar este comando es asignar el contenido del archivo (que está en el disco duro del ordenador) a un objeto que está en la memoria ram y al que hemos denominado `tortugas`. La asignación se realiza a través del operador "`<-`". A partir de ahora, cada vez que utilicemos el término `tortugas` nos estaremos refiriendo al conjunto de datos que acabamos de leer.

NOTA: el objeto `tortugas` es un objeto de clase `tibble` (que podemos leer como "tabla") o `data.frame` (que podemos leer como "base de datos"). Los objetos de esta clase son tablas en las que cada fila representa un objeto (un nido en este caso) y cada columna una variable. Las variables de un mismo `tibble` pueden ser de distinta naturaleza, tanto numéricas como de tipo carácter.

Muchas de las funciones del paquete `tidyverse` están diseñadas para recibir `tibbles` como entrada y devolver `tibbles` como salida.

3. ¿Cuántos nidos tenemos?: `count()`

Comenzamos ahora a utilizar funciones de las librerías del paquete `tidyverse`. La función `count()` cuenta el número de filas del conjunto de datos. Como cada fila es

un nido, `count()` nos proporciona el número de nidos que se han medido en el muestreo.

Para aplicar esta función utilizaremos el operador `pipe` (*tubería*) denotado mediante "`%>%`". La sintaxis "`tortugas %>% count()`" puede leerse como que hacemos pasar a la base de datos `tortugas` a través de una tubería que la lleva hasta `count()` de donde sale el número de filas del archivo:

```
tortugas %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    20
```

Nótese que el resultado de la función `count()` es una tabla (`tibble`).

4. Mejora de la presentación del resultado: `flextable()`

Como acabamos de ver, la función `count()` nos ha devuelto el número de nidos de nuestro muestreo. Sin embargo, la presentación en el documento resultante dista mucho de ser elegante. Si queremos que nuestros resultados tengan buena presentación es preciso aplicarles una capa de "maquillaje". Esto lo podemos hacer mediante la función `flextable()`, del paquete homónimo:

```
tortugas %>% count() %>% flextable()
```

n
20

El paquete flextable permite generar tablas con muy buena presentación, pensadas para que se inserten directamente en documentos html o word.

5. ¿Cuántos nidos tenemos en cada cada playa?: `group_by()`

La utilización de las "tuberías" (`%>%`) junto con las funciones del `tidyverse` permite reutilizar el código de manera muy simple para aplicar una misma función a los datos agrupados según los valores de alguna variable de interés. Así, por ejemplo, si quisiéramos saber cuántos nidos tenemos en cada playa, bastaría emplear el mismo código del punto anterior indicando `group_by(playa)` justo antes de aplicar `count()`:

```
tortugas %>%
  group_by(playa) %>%
  count() %>%
  flextable()
```

playa	n
Calheta	9
Ervatao	4
Ponta Cosme	6
Porto Ferreiro	1

Nótese que en este caso hemos cambiado de línea tras aplicar cada "tubería". El programa funciona igual pero el código es más legible.

6. ¿Cuál es el valor medio de la *Longitud Curva del Caparazón* de las tortugas de la muestra?: `summarise()`

La función `summarise()` es la encargada de aplicar medidas de resumen (media, desviación típica, mediana, asimetrías, etc) a las variables de un `tibble`. Por ejemplo, para calcular el valor medio de la variable LCC (longitud curva del caparazón) procederíamos del siguiente modo:

```
tortugas %>%
  summarise(mean(LCC))
```

```
## # A tibble: 1 x 1
##   `mean(LCC)`
##         <dbl>
## 1        81.6
```

7. ¿Cuánto valen las medias de las variables ACC y Peso?: `summarise()`

Si queremos calcular las medias de dos (o más) variables distintas, bastará con indicarlo dentro del `summarise()`:

```
tortugas %>% summarise(mean(ACC), mean(peso))
```

```
## # A tibble: 1 x 2
##   `mean(ACC)` `mean(peso)`
##         <dbl>        <dbl>
## 1        76.2         59.8
```

8. Calcular de nuevo las tres medias anteriores, mejorando la presentación del resultado: `flextable()` + `autofit()`

Para ello bastará indicar las tres medias que queremos calcular y aplicar `flextable()` a continuación:

```
tortugas %>% summarise(mean(LCC),
                       mean(ACC),
                       mean(peso)) %>%
  flextable()
```

mean(LCC)	mean(ACC)	mean(peso)
81.63	76.24	59.765

Véase que el nombre de las columnas queda dividido en dos lineas. Podemos conseguir que `flextable` ajuste automáticamente el ancho de columna mediante `autofit()`:

```
tortugas %>% summarise(mean(LCC),
                       mean(ACC),
                       mean(peso)) %>%
  flextable() %>%
  autofit()
```

mean(LCC)	mean(ACC)	mean(peso)
81.63	76.24	59.765

9. Modificar la presentación de una tabla `flextable`: tamaño de letra, ancho de columna, título de tabla y decimales.

El paquete `flextble` contiene numerosas funciones para mejorar el aspecto de las tablas. Veremos ahora como modificar cosas como el tamaño de letra, el ancho de columna, el título de la tabla o los decimales de los valores numéricos que contiene.

En los ejemplos anteriores la letra de las tablas resulta un poco pequeña. Podemos hacerla más grande utilizando la función `fontsize()`, indicando si lo que queremos hacer más grande es la cabecera (`part="header"`), el cuerpo de la tabla (`part="body"`) o toda la tabla (`part="all"`):

```
tortugas %>% summarise(mean(LCC),
                       mean(ACC),
                       mean(peso)) %>%
  flextable() %>%
  fontsize(size=14, part="all") %>%
  autofit()
```

mean(LCC)	mean(ACC)	mean(peso)
81.63	76.24	59.765

Si además queremos cambiar el ancho de alguna columna, utilizaremos la función `width`, indicando qué columna(s) queremos modificar y qué ancho queremos poner; por ejemplo, si queremos que el ancho de las columnas 2 y 3 sea 1.5, y el de la columna 1 sea 2:

```
tortugas %>% summarise(mean(LCC),
                       mean(ACC),
                       mean(peso)) %>%
  flextable() %>%
  fontsize(size=14, part="all") %>%
  width(2:3,width=1.5) %>%
  width(1,width=2)
```

mean(LCC)	mean(ACC)	mean(peso)
81.63	76.24	59.765

Podemos añadir fácilmente un título a la tabla mediante la función `set_caption()`. En la sintaxis siguiente, dentro del summarise asignamos un nombre a cada media, y ponemos un titulo:

```
tortugas %>% summarise(LCC=mean(LCC),
                       ACC=mean(ACC),
                       Peso=mean(peso)) %>%
  flextable() %>%
  fontsize(size=14, part="all") %>%
  width(2:3,width=1.5) %>%
  set_caption("Valores medios de las variables morfométricas de las
tortugas")
```

Valores medios de las variables morfométricas de las tortugas

LCC	ACC	Peso
81.63	76.24	59.765

Si en la tabla anterior queremos mostrar los datos solamente con dos decimales utilizamos la función `colformat_num()`. Si no especificamos columnas, se aplica este formato a todas las columnas numéricas:

```
tortugas %>% summarise(LCC=mean(LCC),
                       ACC=mean(ACC),
                       Peso=mean(peso)) %>%
  flextable() %>%
  fontsize(size=14, part="all") %>%
  width(2:3,width=1.5) %>%
  colformat_num(digits=2) %>%
  set_caption("Valores medios de las variables morfométricas de las
tortugas")
```

Valores medios de las variables morfométricas de las tortugas

LCC	ACC	Peso
81.63	76.24	59.77

Por último, si quisiéramos especificar distintos decimales para distintas variables podemos hacerlo del siguiente modo. Nótese que además utilizamos `autofit()` para que ajuste automáticamente el ancho de las columnas:

```
tortugas %>% summarise(LCC=mean(LCC),
                       ACC=mean(ACC),
                       Peso=mean(peso)) %>%
  flextable() %>%
  fontsize(size=14, part="all") %>%
  colformat_num("ACC",digits=2) %>%
  colformat_num(c("LCC","Peso"),digits=4) %>%
  autofit() %>%
  set_caption("Valores medios de las variables morfométricas de las
tortugas")
```

Valores medios de las variables morfométricas de las tortugas

LCC	ACC	Peso
81.6300	76.24	59.7650

10. Calcular los valores medios de las variables anteriores para cada playa: `group_by()`

Nuevamente, si queremos aplicar la misma función a los datos agrupados según los valores de una variable usaremos `group_by()`:

```
tortugas %>%
  group_by(playa) %>%
  summarise(mean(LCC),mean(ACC),mean(peso)) %>%
  flextable() %>%
  fontsize(size=14,part="all") %>%
  autofit()
```

playa	mean(LCC)	mean(ACC)	mean(peso)
Calheta	82.58889	76.67778	62.54444
Ervatao	80.85000	76.77500	56.60000
Ponta Cosme	80.05000	74.91667	57.88333
Porto Ferreiro	85.60000	78.10000	58.70000

Aquí hemos empleado la función `autofit()` que se encarga de ajustar automáticamente el alto y ancho de las filas y columnas de la tabla presentada con `flextable()`.

11. Calcular los valores medios de las variables anteriores para cada año: `group_by()`

```
tortugas %>%
  group_by(Año) %>%
  summarise(mean(LCC), mean(ACC), mean(peso)) %>%
  flextable() %>%
  fontsize(size=14,part="all") %>%
  autofit()
```

Año	mean(LCC)	mean(ACC)	mean(peso)
1999	85.07500	77.70000	63.92500

Año	mean(LCC)	mean(ACC)	mean(peso)
2000	79.35000	76.70000	51.90000
2001	79.86667	74.40000	56.80000
2002	80.62000	74.40000	59.84000
2003	81.60000	75.63333	61.03333
2004	82.03333	79.50000	61.03333