Taller Estadística en R: Sesión 1

Marzo de 2022

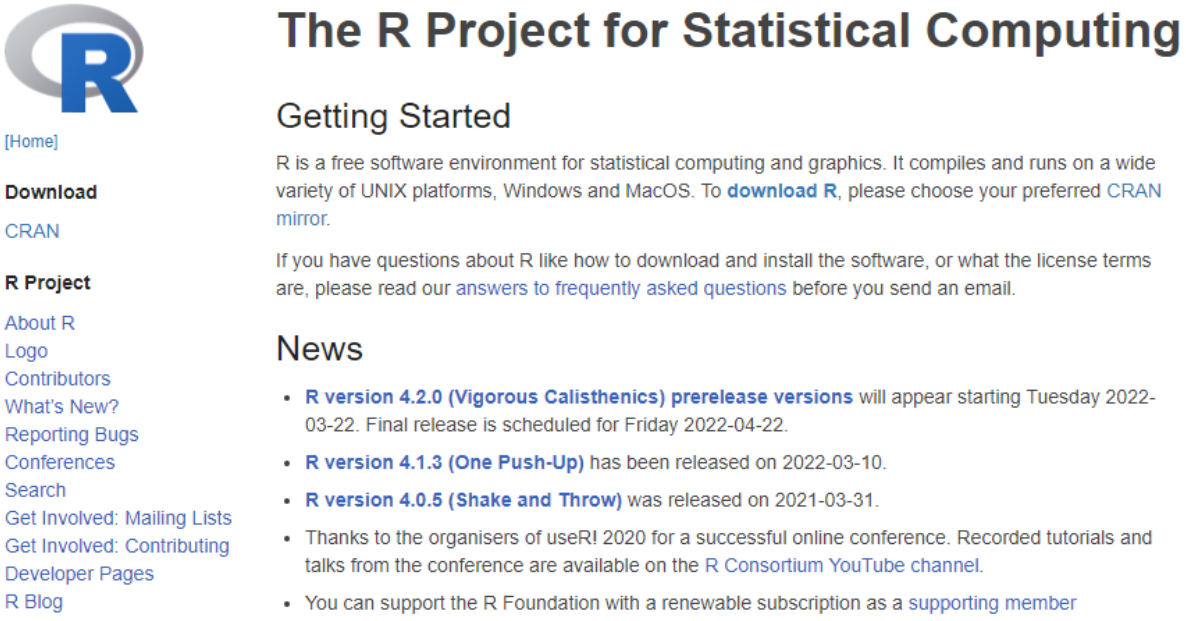
¿Qué es R?

R es un entorno de software libre para computación estadística y gráficos. Compila y se ejecuta en una amplia variedad de plataformas UNIX, Windows y MacOS.

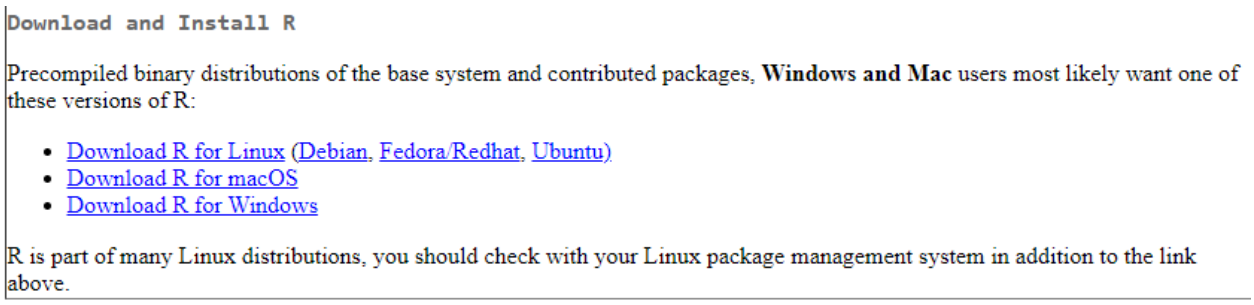
R es un lenguaje de programación que se basa en otro lenguaje llamado S y es principalmente un lenguaje de programación **orientado a objetos**

¿Cómo descargar R?

1. Ir a r-project.org
2. Dar click en download R

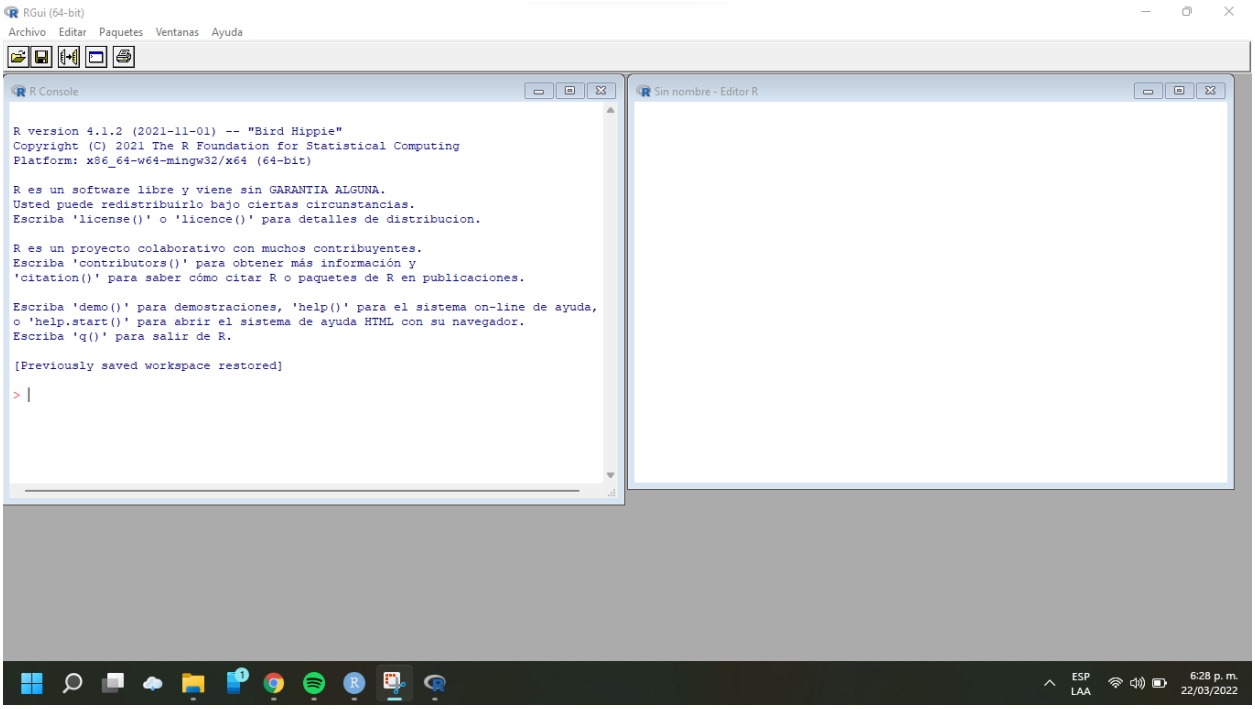


3. Elegir un espejo (Puede ser el de Colombia, aunque es mejor que elijan un espejo de algún país que tenga muchos, como USA) 4. Elegir su sistema operativo y descargar.

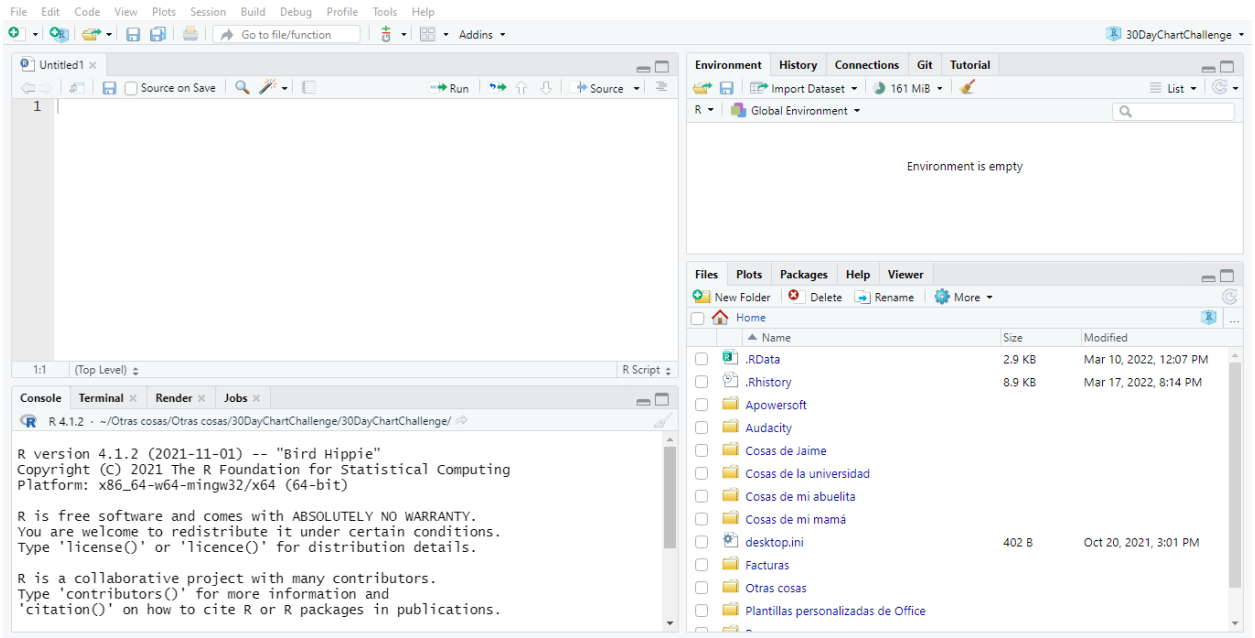


Interfaces gráficas

R puede ser trabajado en la interfaz por default de R, o en alguna interfaz gráfica. Una de las más utilizadas es RStudio.



RStudio es un conjunto de herramientas integradas diseñadas para ayudarlo a ser más productivo con R y Python. Incluye una consola, un editor de resaltado de sintaxis que admite la ejecución directa de código y una variedad de herramientas sólidas para trazar, ver el historial, depurar y administrar su espacio de trabajo.



Hay otras interfaces que permiten usar R como:

- Jupyter
- VSCode

- Tinn-R

Se puede usar cualquiera, y en todas **el resultado debe ser el mismo**, sin embargo la más difundida es RStudio. Para todas estas interfaces R original debe estar descargado, sino no sirven.

Partes de R

Independientemente de si se usa R con o sin interfaz gráfica, siempre hay dos elementos presentes: La consola y el script.

La consola

Es donde se dan todos los procesos computacionales. Ahí es donde se interpreta el código.

El script

Es donde guardamos registro del código. Únicamente guarda el código, es decir, los comandos que utilizamos, pero no guarda el resultado. Lo recomendable es escribir todo en el script e irlo ejecutando, de manera que en el script solo quede el código bueno.

¿Cómo empezar en R?

Paso número 1: Establecer un directorio de trabajo

Lo primero que hay que hacer es establecer un directorio de trabajo, es decir, decirle a R en qué carpeta se encuentran las cosas que necesitamos utilizar en nuestro proyecto y en donde van a quedar guardados los archivos que hagamos.

Para esto es necesario: saber cuál directorio de trabajo está establecido y cambiarlo si es necesario. Para esto vamos a usar dos comandos: getwd() y setwd()

```
getwd() #Nos muestra en qué carpeta tenemos el directorio de trabajo
setwd("C:\\Users\\Maria\\OneDrive")
#nos permite cambiarlo, para cambiarlo hay que poner la dirección entre comillas
#y poner doble backslash (o cambiarlo por un slash)
```

El directorio de trabajo también puede establecerse desde la interfaz gráfica.

Una vez hecho eso podemos empezar a trabajar.

Paso 2: Cargar los paquetes que necesecitemos

Los paquetes son librerías con funciones para desarrollar diferentes tipos de tareas. R trae por defecto instaladas algunas que permiten hacer lo más básico (análisis estadístico simple, gráficos, etc) pero si queremos hacer algo más difícil es posible que necesitemos descargar más paquetes. Para descargar algún paquete de nuestro interés usamos el comando **install.packages()**

```
install.packages("car") #El nombre del paquete debe ir entre comillas
```

Una vez instalado el paquete, para poderlo utilizar debemos llamarlo, para eso usamos el comando **library()**

```
library(car)
```

```
## Loading required package: carData
```

Habiendo hecho esto ya podemos empezar a trabajar.

¿Cómo se trabaja en R?

R es programación orientada a objetos, es decir, toda la información que tengamos se guarda en algo llamado objetos. Para crear objetos utilizamos el operador asignación (<-).

```
x<-1 #el objeto "x" va a adquirir el valor de 1 cuando lo corramos
x
```

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

En los objetos podemos guardar tanto números como caracteres (nombres, palabras) pero los caracteres para ser reconocidos por R deben ir entre comillas, sino R va a pensar que son objetos.

```
nombre<-mariana #Acá va a dar error porque R no encuentra ningún objeto llamado mariana
nombre<-"mariana" #Acá sí nos va a dar
mariana
```

Tipos de objetos

R tiene 5 tipos de objetos esenciales: vectores, listas, data frames, matrices y arrays.

Vectores Son objetos de una única dimensión en donde los elementos pueden ser de diferente naturaleza. Se forman con el comando **c()**

```
edad<-c(12,45,36,59)
#Edades en un grupo de personas, vector de variable numérica
edad
```

```
## [1] 12 45 36 59
```

```
especies<-c("Pitangus sulphuratus","Bubulcus ibis","Troglodytes aedon")
#Especies en una comunidad, vector de variable categórica
#Al ser un vector en donde los elementos son palabras o caracteres
#para que R los reconozca deben estar dentro de comillas
especies
```

```
## [1] "Pitangus sulphuratus" "Bubulcus ibis"        "Troglodytes aedon"
```

Matrices Son objetos de dos dimensiones, donde hay filas y columnas y que generalmente contiene únicamente datos de la misma naturaleza.

Se forman con el comando **matrix()**, el comando tiene 2 argumentos básicos: *ncol* y *nrow*, *ncol* especifica el número de columnas y *nrow* el de filas.

Se pueden llenar con vectores o con los datos individuales

```
matriz1<-matrix(c("a","b","c","d"),ncol = 2,nrow = 2)
matriz1
```

```
##      [,1] [,2]
## [1,] "a"  "c"
## [2,] "b"  "d"
```

```
vector1<-c("x","y")
vector2<-c("z","w")
matriz2<-matrix(c(vector1,vector2),ncol = 2,nrow = 2)
matriz2
```

```
##      [,1] [,2]
## [1,] "x"  "z"
## [2,] "y"  "w"
```

Otro argumento importante es *byrow*, que determina como R llena la matriz, este argumentos es *lógico*, es decir, solo hay dos posibles maneras de llenarlo: TRUE o FALSE. TRUE significa que la matriz se va a llenar llenando las filas, FALSE significa que la matriz se va a llenar llenando las columnas.

```
matriz3<-matrix(1:9,ncol=3,nrow=3, byrow = TRUE)
matriz3
```

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

```
matriz4<-matrix(1:9,ncol=3,nrow=3, byrow = FALSE)
matriz4
```

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

Data frames Un data frame es la estructura que se usa en R para los ficheros de datos correspondientes a una serie de variables medidas en cada sujeto o unidad bajo estudio. Un data frame es como una matriz con entradas que pueden ser de distintos tipos (números, palabras, etc.)

El comando para formarlos es **data.frame()**

```
nombres<-c("Alejandro","Sebastián","Mariana")
edad<-c(24,23,21)
dataframe1<-data.frame(nombres,edad)
dataframe1
```

```
##     nombres edad
## 1 Alejandro   24
## 2 Sebastián   23
## 3   Mariana   21
```

Si quiero añadir columnas o filas a un data frame o una matriz puedo utilizar los comando **cbind** y **rbind**

```
sexo<-c("M","M","F")
dataframe2<-cbind(dataframe1,sexo)
dataframe2
```

```
##     nombres edad sexo
## 1 Alejandro   24    M
## 2 Sebastián   23    M
## 3   Mariana   21    F
```

```
matriz5<-rbind(matriz3,matriz4)
matriz5
```

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

Listas Es parecido a un vector, sin embargo, dentro de ella puede contener varios objetos, donde cada objeto puede ser de un tipo diferente.

Se forman con el comando **list()**

```
lista<-list(a=dataframe1,b=matriz5,c=vector1)
lista #en vez de "a", "b" y "c" puedo poner lo que yo quiera
```

```
## $a
##     nombres edad
## 1 Alejandro   24
## 2 Sebastián   23
## 3   Mariana   21
##
## $b
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]    1    4    7
## [5,]    2    5    8
## [6,]    3    6    9
##
## $c
## [1] "x" "y"
```

Arrays Un array es una estructura de datos que puede contener datos multidimensionales. En R, los arrays son objetos que pueden contener dos o más datos bidimensionales. Por ejemplo, en matrices cuadradas pueden contener dos filas y dos columnas y la dimensión puede tomar cinco.

Esto solo lo usa la gente muy tesa y por eso no lo vamos a profundizar

Como ya conocemos los tipos de objetos ya podemos hacer operaciones con ellos.

Operaciones básicas en R

En R pueden hacerse todas las operaciones básicas, tanto sobre los objetos como sobre números que no hayamos guardado

```
1+1 #suma normal
```

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

```
m<-c(2,4)
m
```

```
## [1] 2 4
```

```
m*4 #operacion en el vector
```

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

```
n<-c(3,5)

n+m #también se pueden hacer operaciones entre vectores
```

```
## [1] 5 9
```

```
log(matriz3) #operación en la matriz
```

```
##          [,1]      [,2]     [,3]
## [1,] 0.000000 0.6931472 1.098612
## [2,] 1.386294 1.6094379 1.791759
## [3,] 1.945910 2.0794415 2.197225
```

```
matriz3*matriz4 #operación entre matrices
```

```
##      [,1] [,2] [,3]
## [1,]    1    8   21
## [2,]    8   25   48
## [3,]   21   48   81
```

Operaciones básicas

- Sumas (usando el +)
- Restas (usando el -)
- Potencias (usando el ^)
- Raíz cuadrada (usando el comando **sqrt()**)
- Logaritmo (usando el comando **log()**)

Otras operaciones útiles

Crear secuencias En R se pueden crear secuencias de números usando el comando **seq()**. El comando se llena de la siguiente manera:

seq(número donde inicia la secuencia, número donde acaba la secuencia, cada cuantos números)

```
secuencia<-seq(1,20,2) #Una secuencia que empiece en 1, termine en 20, y sea cada 2 números
secuencia
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

Secuencias de números consecutivos también se pueden crear usando :

```
secuencia2<-1:10 #secuencia con los números del 1 al 10
```

Repeticiones A veces es necesario repetir varias veces los mismos datos, por lo que es más fácil usar el comando **rep()** que copiar a mano. El comando se usa: rep(lo que voy a repetir, cuantas veces lo voy a repetir)

```
repeticion<-c(rep("Mariana",3))
repeticion
```

```
## [1] "Mariana" "Mariana" "Mariana"
```

Indexar Indexar es el proceso de extraer los datos de un objeto utilizando su posición dentro del objeto. Esto se hace utilizando los corchetes.

```
semillero<-c("dany","mariana","ana")
semillero[2] # qué hay en la posición 2 del vector semillero
```

```
## [1] "mariana"
```

```
matriz2[1,2] #qué hay en celda de la fila 1, columna 2 de la matriz2
```

```
## [1] "z"
```

Mirar cuán largo es un objeto Para mirar cuantos elementos tiene un vector podemos utilizar el comando **length()** y para mirar cuantas filas y columnas tiene un data frame o una matriz podemos utilizar **dim()**

```
length(semillero)
```

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

```
dim(matriz5)
```

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

```
dim(dataframe1)
```

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

Importar datos

La mayoría de veces ya tenemos los datos como otro tipo de archivos y no necesitamos introducir dato por dato en R, por eso se hace necesario importarlos. En R se pueden importar datos de múltiples formatos sin problema. Los comandos básicos para hacerlo son:

- read.table()
- read.delim() (si read.table no les da, generalmente este sí)
- read.csv() (este, como su nombre lo dice, solo sirve para archivos de tipo comma separated values)
- read_excel (este requiere la instalación del paquete *readxl* y solo sirve para archivos con extensión xlsx)

Todos estos comandos tienen la misma estructura:

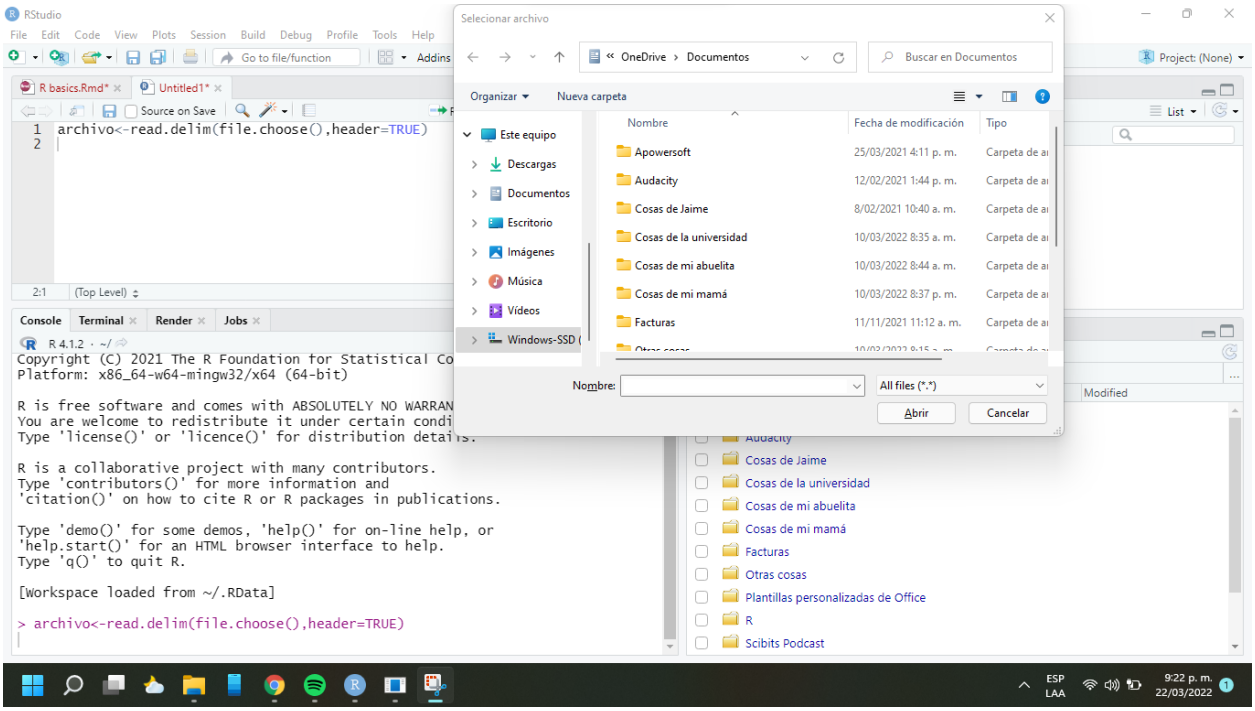
read.table("nombre del archivo.extensión de archivo",header=argumento de tipo lógico, si es TRUE significa que la primera fila se toma como los nombres de las variables)

Si vamos a subir un archivo que está en una carpeta diferente a la que establecimos como directorio de trabajo *tenemos que copiar la ruta completa del archivo**

```
datos<-read.table("Limia caymanensis.txt",header = TRUE)
head(datos)
```

```
##   Fish Sex Length AnalFinLength DorsalFinLength DorsalFinHeight
## 1   F5   F  22.54          2.91            2.60            2.01
## 2   F6   F  22.60          3.16            2.73            2.70
## 3   F7   F  23.12          3.21            3.01            2.05
## 4   F4   F  23.64          3.05            2.81            1.92
## 5   F2   F  23.92          3.16            3.23            2.40
## 6   F1   F  24.45          2.95            3.30            2.47
##   BodyDepthMidBody ln.Length. ln.AnalFinLength. ln.DorsalFinLength.
## 1             8.93   3.115292          1.068153           0.9555114
## 2             8.50   3.117950          1.150572           1.0043016
## 3             9.29   3.140698          1.166271           1.1019401
## 4             9.60   3.162940          1.115142           1.0331845
## 5            10.00   3.174715          1.150572           1.1724821
## 6             9.49   3.196630          1.081805           1.1939225
##   ln.DorsalFinHeight. ln.BodyDepth.
## 1           0.6981347      2.189416
## 2           0.9932518      2.140066
## 3           0.7178398      2.228939
## 4           0.6523252      2.261763
## 5           0.8754687      2.302585
## 6           0.9042182      2.250239
```

Podemos usar el argumento file.choose() dentro de cualquiera de esos comandos, este argumento nos abre un cuadro para buscar nuestro archivo dentro de nuestros documentos



Pedir ayuda en R

Si no saben cómo se usa una función, qué función sirve para lo que ustedes quieren hacer o qué funciones trae un paquete, pueden pedirle ayuda a R.

Hay comandos que les ayudan a investigar sobre lo que necesitan hacer en R, algunos son:

- **help()**: Sirve para buscar ayuda sobre comandos específicos
- **help.search()**: Sirve para buscar ayuda sobre temas más generales
- **apropos()**: Sirve para ver qué comandos están relacionados con el comando que puse