Introducción a R

Semillero de investigación en Bioestadística

RRCJ-2019

Contents

Inicio.

Instaladores.

En esta guía encontrará una breve descripción de algunos aspectos básico del programa **R** que le permitirán comenzar a utilizar la herramienta. Primero debe instalar los dos programa: **R** y **RStudio**; **RStudio** se conoce como un **entorno de desarrollo integrado** (IDE: integrated development environment) que a partir de un ambiente que incluye múltiples herramientas, facilita el manejo del programa **R**. Es decir, que utilizaremos **R** a través de **RStudio**.

En los siguientes enlaces se descargan los instaladores según el sistema operativo que utilice:

- R (Windows):

https://cran.r-project.org/bin/windows/base/

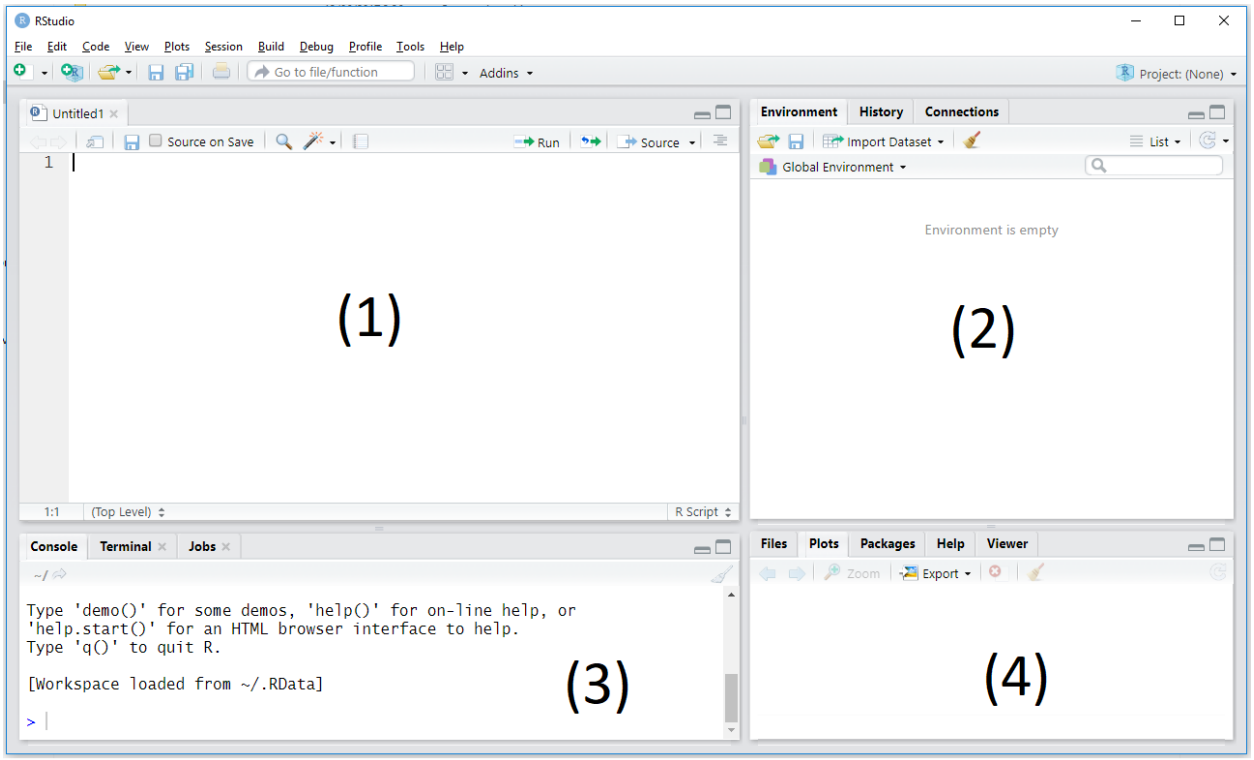
- R (Mac):

https://cran.r-project.org/

- RStudio (Windows y Mac):

https://www.rstudio.com/products/rstudio/download/#download

Después de instalar los dos programas, abra **RStudio** donde encontrará los siguientes cuatro paneles:



En el panel (1) se escriben las instrucciones o comandos requeridos para el análisis de un conjunto de datos, en el panel (2) se encuentran, principalmente, los datos que se piensan utilizar organizados en distintos "objetos" (los que revisaremos más adelante), en el panel (3) se presentan los resultados correspondientes a la corrida de un comando específico y en el panel (4) se presentan, entre otras cosas, los resultados de gráficos y ayudas de los comandos que se utilicen.

El directorio de trabajo

Cuando inicia una fase de análisis utilizando **R**, se acostumbra definir una **carpeta de trabajo** donde se guardan todos los archivos relacionados con el proyecto (datos originales, archivos de código, imágenes y gráficos, étc.)

Para definir este **directorio de trabajo**, se escribe la siguiente instrucción en el panel (1):

```
setwd("C:/Users/introducción")
```

y después, se coloca el cursor en cualquier parte de este código y se teclea `Ctrl+Enter` (en Windows) o `Command+Enter` (en Mac).

La acción anterior es la forma de correr o ejecutar un código en **RStudio**.

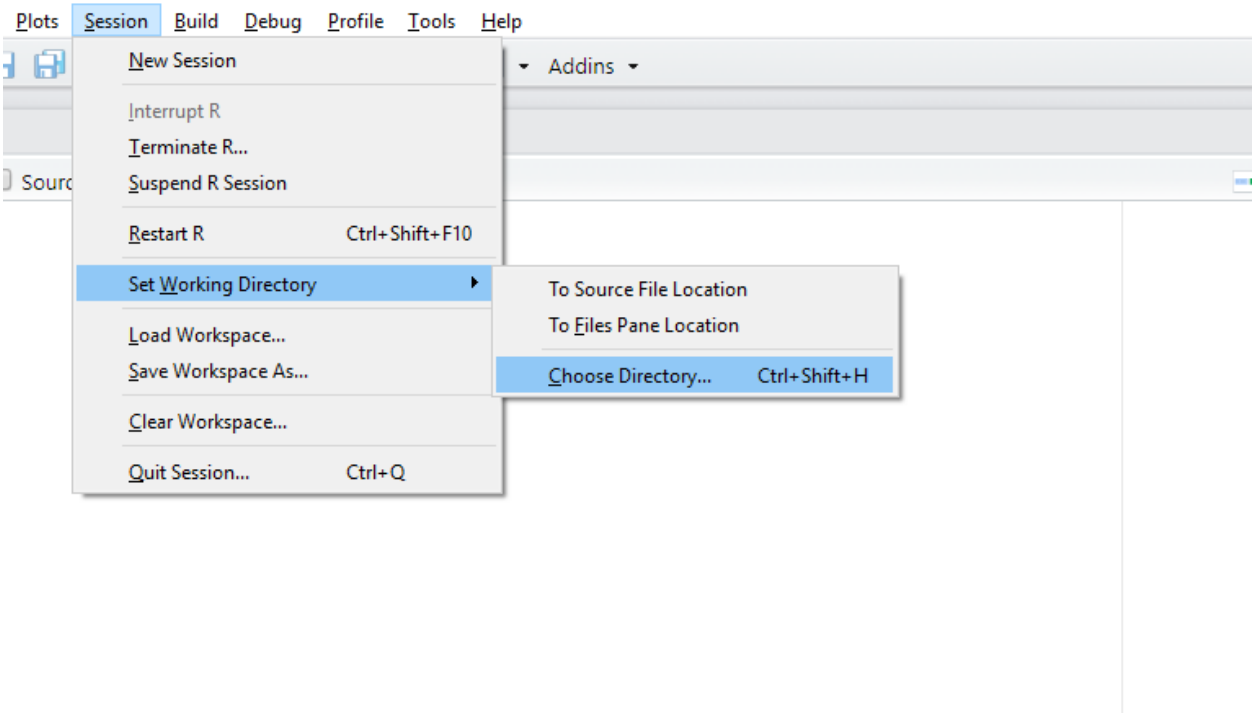
Si el **directorio de trabajo** ya está definido, y se quiere conocer su ruta o ubicación, se corre la siguiente instrucción:

```
getwd()
```

Para conocer que archivos se encuentran en el **directorio de trabajo** se utiliza el siguiente código:

```
dir()
```

El **directorio de trabajo** también se puede definir por ventanas.



Paquetes y funciones

Algunas instrucciones ya se encuentran instaladas en el programa y se pueden utilizar directamente. Estas instrucciones se denominan **funciones** y un conjuntos de funciones se denominan **paquetes**. Para ver un ejemplo de como utilizar una **función** ya instalada, primero tabulemos 10 datos de edades de la siguiente forma en el panel (1). Ejecutemos este código:

```
edad<-c(12,14,15,14,16,13,18,17,19,15)
edad
```

```
##  [1] 12 14 15 14 16 13 18 17 19 15
```

> Note que la forma de asignar un conjunto de valores a un nombre especifico es con el símbolo: `<-`
> Adicionalmente, en el panel (2), apareció el objeto **edad** acorde al nombre que le asignamos.
> Cada vez que se necesiten estos valores, basta con escribir **edad**.

Ahora, para calcular por ejemplo la media de estas edades, utilizaremos la **función** `mean` del **paquete** {base}. Corramos estas función:

```
mean(edad,trim = 0.1,na.rm=TRUE)
```

```
## [1] 15.25
```

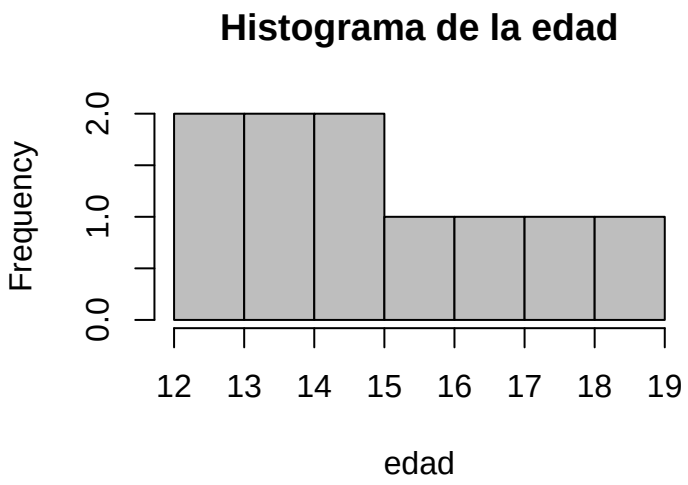
Para conocer a que corresponden los resultados obtenidos de una función, se utiliza la **ayuda** de la siguiente forma:

```
?mean
```

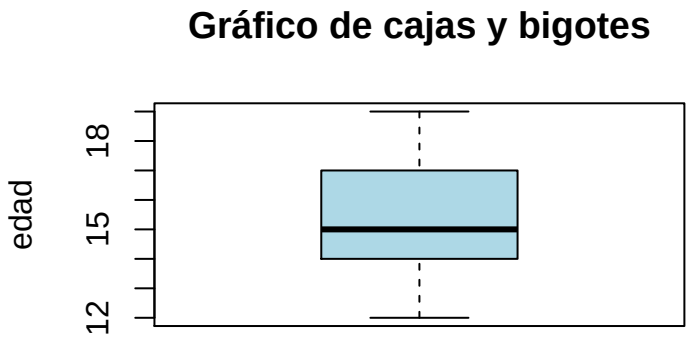
la documentación de la función aparece en el panel (4).

Veamos algunos ejemplos de otras **funciones** ya pre-instaladas, utilizando los 10 datos de la edad.

```
hist(edad,main = "Histograma de la edad",xlab = "edad",col = "grey")
```




```
boxplot(edad, main="Gráfico de cajas y bigotes",ylab="edad",col="lightblue")
```



Se pueden consultar que **paquetes** ya se encuentran instalados corriendo el siguiente código:

```
library()
```

o se pueden ver en el panel (4) en la pestaña "Packages".

Se pueden instalar nuevos paquetes con funciones adicionales en **R**. Para instalar un nuevo paquete, se debe conocer su nombre y colocarlo dentro de la siguiente función:

```
install.packages("nombre del paquete")
```

Por ejemplo, para instalar el paquete `cowsay` se corre el siguiente código:

```
install.packages("cowsay")
```

Ahora, después de haber finalizado la instalado el paquete, debemos "llamarlo" con la instrucción `library()` para poderlo utilizar; así:

```
library(cowsay)
```

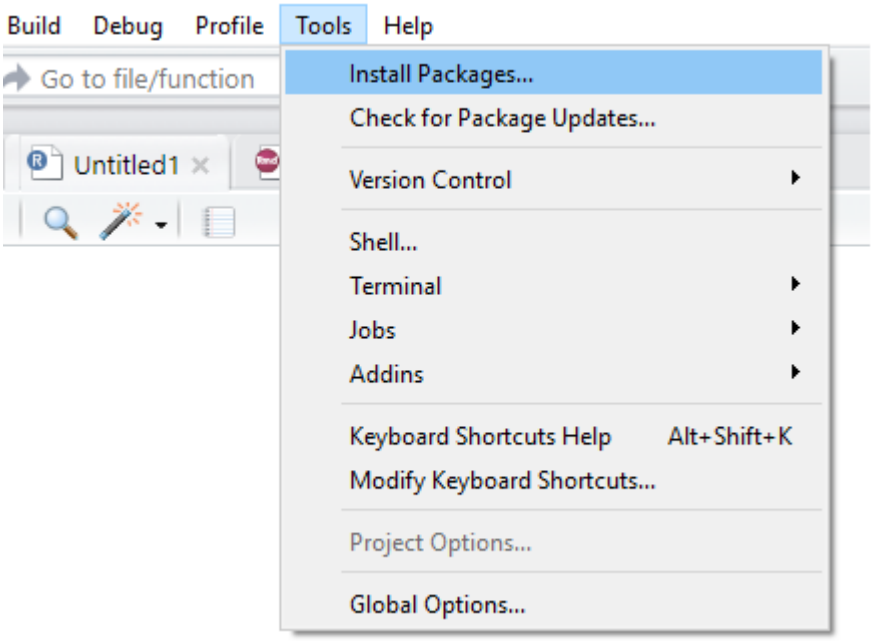
ahora si, podemos utilizar la función `say` de este paquete corriendo el siguiente código:

```
say("Hola mundo!", by="cow")

# cow, chicken, clippy, poop, bigcat, ant, pumpkin, ghost,
# spider, rabbit, pig, snowman, frog, hypnotoad, facecat,
# behindcat, cat, trilobite, shark, buffalo, smallcat, yoda,
# endlesshorse, bat, bat2, turkey, monkey, daemon,
# egret, duckling, duck, owl, rms, random
```

> El texto que se escribe después del signo # corresponde a documentación y no afecta la ejecución del código.

También realizar la instalación de nuevos paquetes por ventanas en: **Tools > Install Packages**, así:



Objetos en R

Existen distintos tipos de arreglos de datos, que llamaremos **objetos**, que se pueden crear y utilizar en el programa. A continuación se describen.

Vectores

Un vector corresponde a un conjunto de valores con la misma estructura.

Por ejemplo, creemos 4 vectores (como **edad** anteriormente) corriendo el siguiente código:

```
peso<-c(60,54,55,62,75,64,58,69,71,52)
talla<-c(1.74,1.68,1.59,1.6,1.78,1.65,1.75,1.69,1.71,1.59)
sexo<-c("H","M","M","H","H","M","M","H","M","M")
```

```
educacion<-c("secundaria","universitario","posgrado","universitario",
       "universitario","primaria","secundaria","universitario",
       "secundaria","posgrado")
```

los dos primeros tienen una estructura con valores numéricos y los dos últimos con caracteres.

La estructura de un vector se puede consultar así:

```
str(peso)
```

```
##  num [1:10] 60 54 55 62 75 64 58 69 71 52
```

aunque esta información ya se presenta en el panel (2).

También se pueden crear vectores utilizando la función `rep`:

```
id<-rep(1:10);id
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
pais<-rep("COL",time=10);pais
```

```
##  [1] "COL" "COL" "COL" "COL" "COL" "COL" "COL" "COL" "COL" "COL"
```

```
depto<-rep(c("BOG","ATL"),each=5);depto
```

```
##  [1] "BOG" "BOG" "BOG" "BOG" "BOG" "ATL" "ATL" "ATL" "ATL" "ATL"
```

Podemos consultar la longitud (cantidad de valores) y cuales son los distintos valores de un vector:

```
length(educacion)
```

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

```
unique(educacion)
```

```
## [1] "secundaria"    "universitario" "posgrado"      "primaria"
```

Si queremos cambiar la estructura de un vector y guardarlo con otro nombre:

```
educacion_ord<-factor(educacion,levels = c("primaria","secundaria",
                                           "universitario","posgrado"))

str(educacion_ord)
```

```
##  Factor w/ 4 levels "primaria","secundaria",..: 2 3 4 3 3 1 2 3 2 4
```

```
educacion_num<-as.numeric(educacion_ord)
educacion_num
```

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

```
educacion_carac<-as.character(educacion_num)
educacion_carac
```

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

borremos los tres vectores anteriores:

```
rm(educacion_ord,educacion_num,educacion_carac)
```

Reportar valores ubicados en una posición específicos dentro de un vector o cambiar este valor:

```
sexo[4]
```

```
## [1] "H"
```

```
# cambiar un valor en una posición específica:

edad
```

```
##  [1] 12 14 15 14 16 13 18 17 19 15
```

```
edad[9]<-21
edad
```

```
##  [1] 12 14 15 14 16 13 18 17 21 15
```

Ahora, veamos algunas operaciones básicas entre vectores:

```
edad+peso-talla # sumas y restas
```

```
##  [1] 70.26 66.32 68.41 74.40 89.22 75.35 74.25 84.31 90.29 65.41
```

```
edad*talla # multiplicación
```

```
##  [1] 20.88 23.52 23.85 22.40 28.48 21.45 31.50 28.73 35.91 23.85
```

```
sqrt(talla) # raiz cuadrada
```

```
##  [1] 1.319091 1.296148 1.260952 1.264911 1.334166 1.284523 1.322876
##  [8] 1.300000 1.307670 1.260952
```

```
peso/talla^2 # cociente y potencia
```

```
##  [1] 19.81768 19.13265 21.75547 24.21875 23.67125 23.50781 18.93878
##  [8] 24.15882 24.28098 20.56881
```

Matrices

Son un objeto que tiene dos dimensiones, donde la primera determina el número de filas y la segunda el número de columnas. Se pueden crear matrices a partir de los vectores anteriores, así:

```
datos<-matrix(c(edad,peso,talla),nrow = 10,ncol = 3,
              dimnames = list(1:10,c("edad","peso","talla")))

datos
```

```
##    edad peso talla
## 1    12   60  1.74
## 2    14   54  1.68
## 3    15   55  1.59
## 4    14   62  1.60
## 5    16   75  1.78
## 6    13   64  1.65
## 7    18   58  1.75
## 8    17   69  1.69
## 9    21   71  1.71
## 10   15   52  1.59
```

Consultar valores específicos dentro de una matriz:

```
datos[4,2]
```

```
## [1] 62
```

```
datos[6,]
```

```
##  edad  peso talla
## 13.00 64.00  1.65
```

```
datos[,3]
```

```
##    1    2    3    4    5    6    7    8    9   10
## 1.74 1.68 1.59 1.60 1.78 1.65 1.75 1.69 1.71 1.59
```

transponer esta matriz:

```
datos2<-t(datos)
datos2
```

```
##           1     2     3    4     5     6     7     8     9    10
## edad  12.00 14.00 15.00 14.0 16.00 13.00 18.00 17.00 21.00 15.00
## peso  60.00 54.00 55.00 62.0 75.00 64.00 58.00 69.00 71.00 52.00
## talla  1.74  1.68  1.59  1.6  1.78  1.65  1.75  1.69  1.71  1.59
```

Ahora, construimos una matriz adicionando el vector sexo y consultamos su estructura:

```
datos3<-matrix(c(edad,peso,talla,sexo),nrow = 10,ncol = 4,
               dimnames = list(1:10,c("edad","peso","talla","sexo")))


datos3
```

```
##    edad peso talla  sexo
## 1  "12" "60" "1.74" "H"
## 2  "14" "54" "1.68" "M"
## 3  "15" "55" "1.59" "M"
## 4  "14" "62" "1.6"  "H"
## 5  "16" "75" "1.78" "H"
## 6  "13" "64" "1.65" "M"
## 7  "18" "58" "1.75" "M"
## 8  "17" "69" "1.69" "H"
## 9  "21" "71" "1.71" "M"
## 10 "15" "52" "1.59" "M"
```

```
str(datos3)
```

```
##  chr [1:10, 1:4] "12" "14" "15" "14" "16" "13" "18" "17" "21" "15" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:10] "1" "2" "3" "4" ...
##   ..$ : chr [1:4] "edad" "peso" "talla" "sexo"
```

lo anterior muestra, que todos los valores dentro de una matriz también tienen la misma estructura.

data.frame

Este objeto permite reunir vectores con diferentes estructuras. Creemos un **data.frame** a partir de los vectores iniciales y veamos su estructura, así:

```
bd<-data.frame(edad,peso,talla,sexo)
bd
```

```
##    edad peso talla sexo
## 1    12   60  1.74    H
```

```
## 2    14   54  1.68    M
## 3    15   55  1.59    M
## 4    14   62  1.60    H
## 5    16   75  1.78    H
## 6    13   64  1.65    M
## 7    18   58  1.75    M
## 8    17   69  1.69    H
## 9    21   71  1.71    M
## 10   15   52  1.59    M
```

```
str(bd)
```

```
## 'data.frame':    10 obs. of  4 variables:
##  $ edad : num  12 14 15 14 16 13 18 17 21 15
##  $ peso : num  60 54 55 62 75 64 58 69 71 52
##  $ talla: num  1.74 1.68 1.59 1.6 1.78 1.65 1.75 1.69 1.71 1.59
##  $ sexo : Factor w/ 2 levels "H","M": 1 2 2 1 1 2 2 1 2 2
```

Para identificar información específica dentro de un **data.frame**:

```
bd$edad
```

```
##  [1] 12 14 15 14 16 13 18 17 21 15
```

```
bd$sexo[1]
```

```
## [1] H
## Levels: H M
```

```
bd[1,4]
```

```
## [1] H
## Levels: H M
```

```
bd[5,]
```

```
##   edad peso talla sexo
## 5   16   75  1.78    H
```

Guandemos todos los valores distintos que toma la talla clasificada por sexo:

```
talla_sexo<-by(bd,bd$sexo,function(bd)unique(bd$talla));

talla_sexo$H
```

```
## [1] 1.74 1.60 1.78 1.69
```

```
talla_sexo$M
```

```
## [1] 1.68 1.59 1.65 1.75 1.71
```

Obtener el número de valores distintos de la talla por sexo:

```
length(talla_sexo$H)
```

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

```
length(talla_sexo$M)
```

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

Listas

Dentro de una lista podemos colocar todos los objetos descritos anteriormente:

```
varios<-list(peso,datos,bd)
```

Ubicar información específica dentro de una lista:

```
varios[[1]]
```

```
##  [1] 60 54 55 62 75 64 58 69 71 52
```

```
varios[[1]][5]
```

```
## [1] 75
```

```
varios[[2]]
```

```
##    edad peso talla
## 1    12   60  1.74
## 2    14   54  1.68
## 3    15   55  1.59
## 4    14   62  1.60
## 5    16   75  1.78
## 6    13   64  1.65
## 7    18   58  1.75
## 8    17   69  1.69
## 9    21   71  1.71
## 10   15   52  1.59
```

```
varios[[2]][2,3]
```

```
## [1] 1.68
```

```
varios[[3]]
```

```
##    edad peso talla sexo
## 1    12   60  1.74    H
## 2    14   54  1.68    M
## 3    15   55  1.59    M
## 4    14   62  1.60    H
## 5    16   75  1.78    H
## 6    13   64  1.65    M
## 7    18   58  1.75    M
## 8    17   69  1.69    H
## 9    21   71  1.71    M
## 10   15   52  1.59    M
```

```
varios[[3]]$peso
```

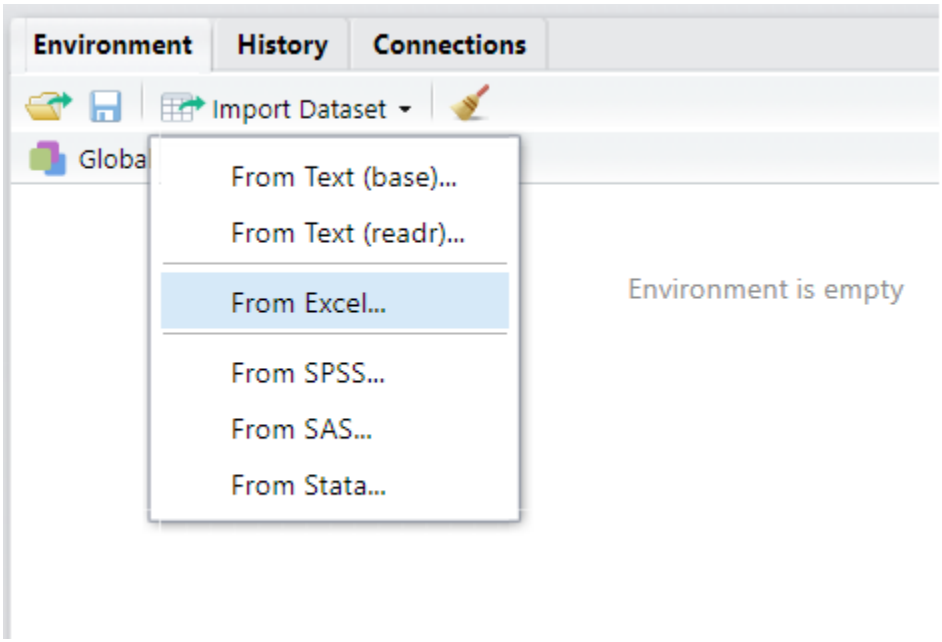
```
##  [1] 60 54 55 62 75 64 58 69 71 52
```

```
varios[[3]]$peso[5]
```

```
## [1] 75
```

Importar datos

RStudio tiene ventanas (en el panel (4)) para realizar la importación de datos que se encuentren en otros programas tales como Excel, SPSS, SAS o Stata.



También existen paquetes que se pueden instalar que cumplen este propósito. El paquete `rio` con su función `import` es uno de estos.

Actividad

1. Cree 2 vectores con 16 números cada uno.
2. Cree 2 vectores con 16 caracteres cada uno.
3. Utilizando la función `rep`, cree un vector con los siguientes valores:
   - "A" "A" "B" "B" "A" "A" "B" "B" "A" "A" "B" "B" "A" "A" "B" "B"

Apóyese en la ayuda de la función.

4. ¿Cuántos valores diferentes tiene cada uno de los vectores tipo caracter que creo?
5. A uno de los vectores con estructura número, cámbielo a una estructura caracter.
6. A uno de los vectores con estructura caracter, cámbielo a una estructura tipo factor.
7. Coloque el valor "casa" en la posición 10 de uno de los vectores tipo caracter.
8. Cree una matriz con los dos vectores numéricos y los dos vectores tipo caracter.
9. ¿Qué valor hay en la fila 2 y en la columna 3?, ¿Qué valores hay en la fila 6?.
10. Cree un data.frame con los 4 vectores creados al inicio de la actividad.
11. ¿Qué valor hay en el tercer vector en la posición 13?
12. Instale el paquete `gmodels`.
13. ¿Qué hace la función `CrossTable` de este paquete?.

14. Use la función `CrossTable` con las dos variables tipo caracter incluidas en el data.frame.
15. Cree una lista con dos vectores, una matriz y un data.frame.
16. Reporte la posición 8 del segundo vector incluido en la lista.
17. Reporte el valor en la fila 3 y la columna 2 de la matriz guardada en la lista.
18. Reporte la segunda columna del data.frame incluido en la lista.