eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Métodos Estadísticos en Ingeniería

Práctica 1. Primeros pasos en *R con RStudio*

¿Qué es R?

R es un proyecto de software libre de GNU. GNU es un **proyecto colaborativo de sofware libre** auspiciado por la Free Software Foundation. R es un lenguaje de programación que cuenta con una comunidad extensa de desarrolladores, investigadores y usuarios. **Es un lenguaje orientado a objetos. Las variables, datos, funciones, resultados, etc., se guardan en la memoria del computador en forma de Objeto con un nombre específico.** El usuario puede **modificar o manipular estos objetos con operadores** (aritméticos, lógicos, y comparativos). A la vez, R es un entorno de trabajo con una serie de utilidades para manipulación de datos, cálculo y representación gráfica. Está disponible para distintos sistemas operativos de tipo Unix y similares (FreeBSD y Linux), Windows y Mac OS.

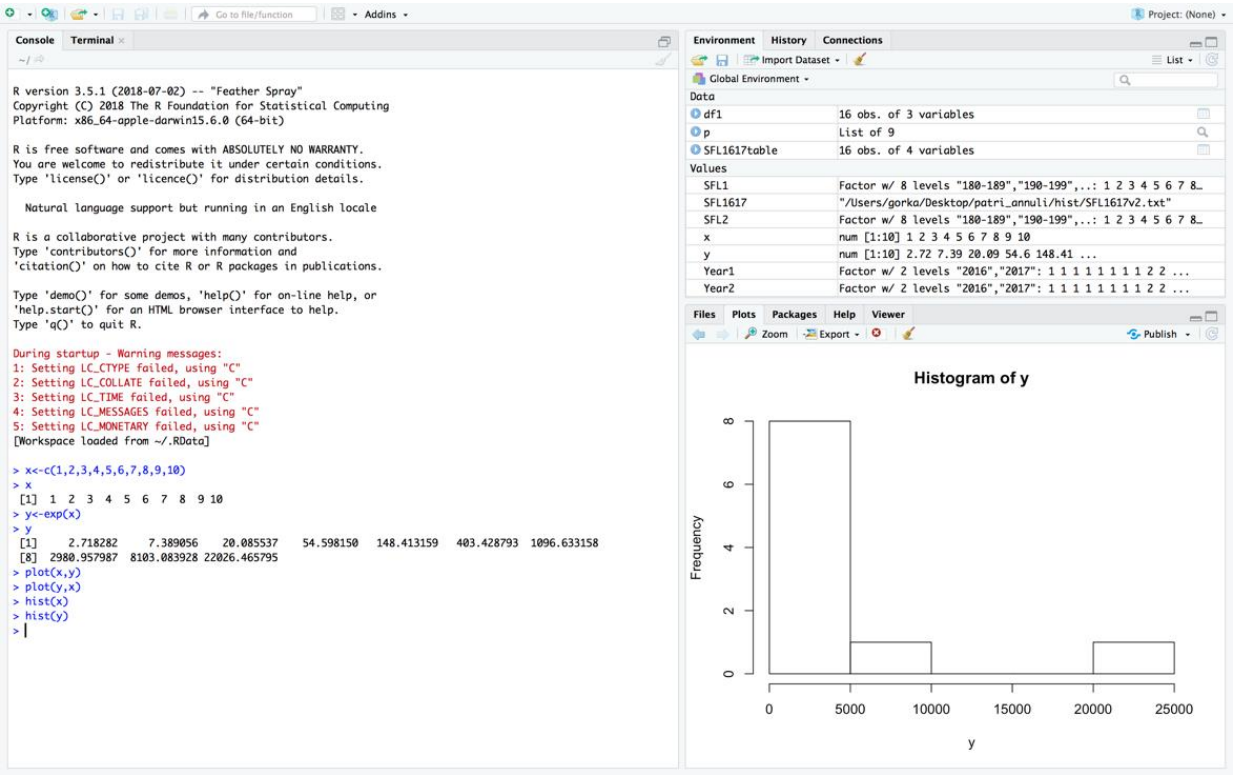
¿Qué es RStudio?

RStudio es el entorno de trabajo y edición que utilizaremos, mucho más amigable que Rconsole o Rcommander. Tendremos que R instalado anteriormente para trabajar con RStudio. Es un entorno de desarrollo integrado (IDE) para R. Es software libre y se puede ejecutar sobre distintas plataformas (Windows, Mac, or Linux) o incluso desde la web usando RStudio Server.

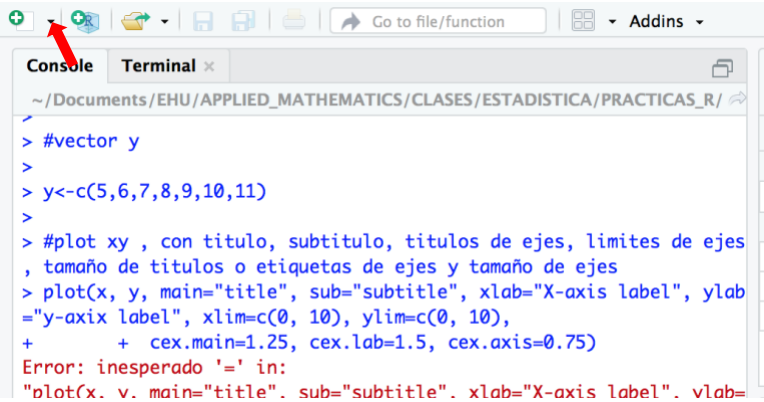
Entraremos en **RStudio**. Clickando en el icono de RStudio en el escritorio. Automáticamente se carga R. En RStudio, en pantalla, aparte de la línea superior de menús aparecen cuatro ventanas. Aunque la primera vez que lo abramos pueden aparecer tres, siendo, en primera instancia, la superior izquierda, la consola (Console):

VENTANAS DE R STUDIO

Inicialmente tenemos 3 ventanas: (1) Console, (2) Environment/History, (3) Packages/Plots/Help.

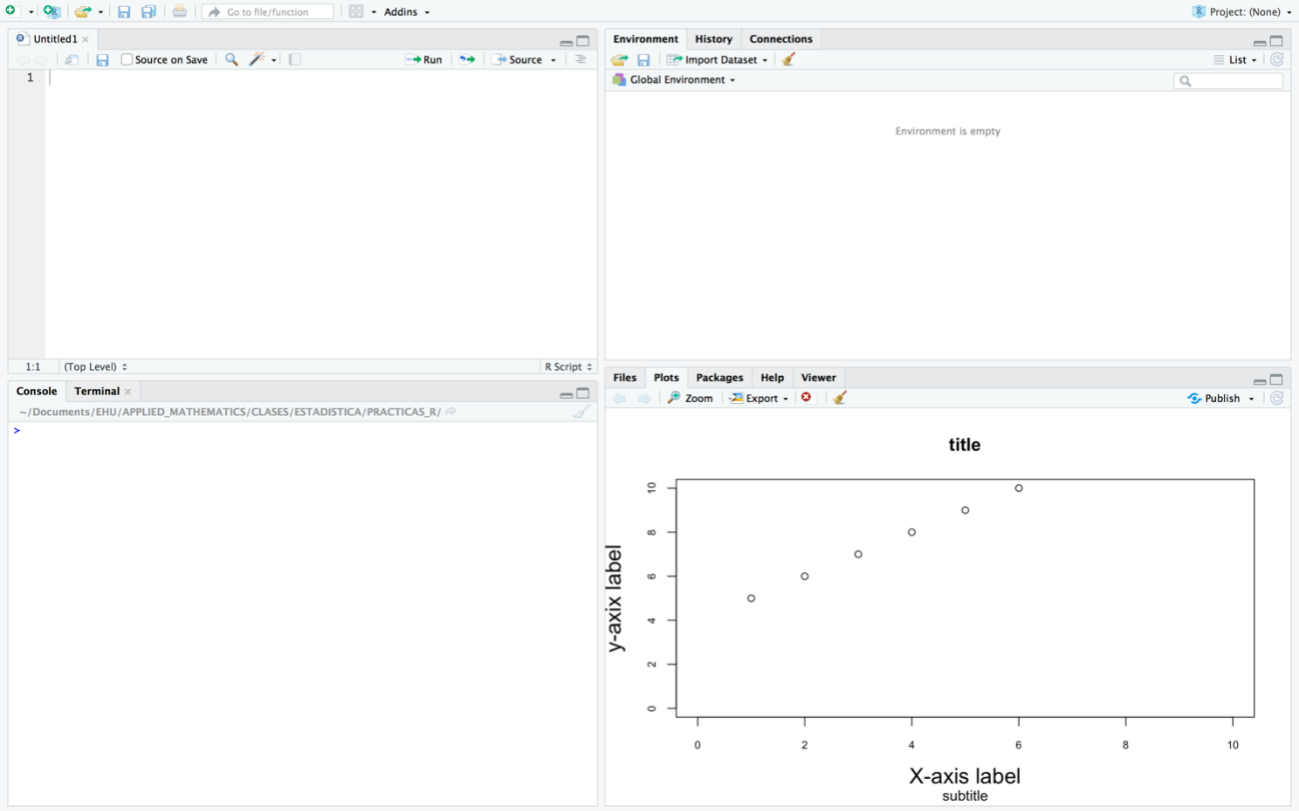


Para abrir la cuarta ventana (Scripts) hacemos click en la parte superior izquierda de la consola (Console), concretamente en la flecha para abajo a la derecha del simbolo en verde +.



Entramos y clickamos en Rscript. Nos abre una ventana nueva en la parte superior izquierda donde se abren los scripts. En este caso tenemos abierto el primer script o archivo para edición de códigos. Lo veremos con el nombre Untitled1 y la extensión típica de los scripts .R.

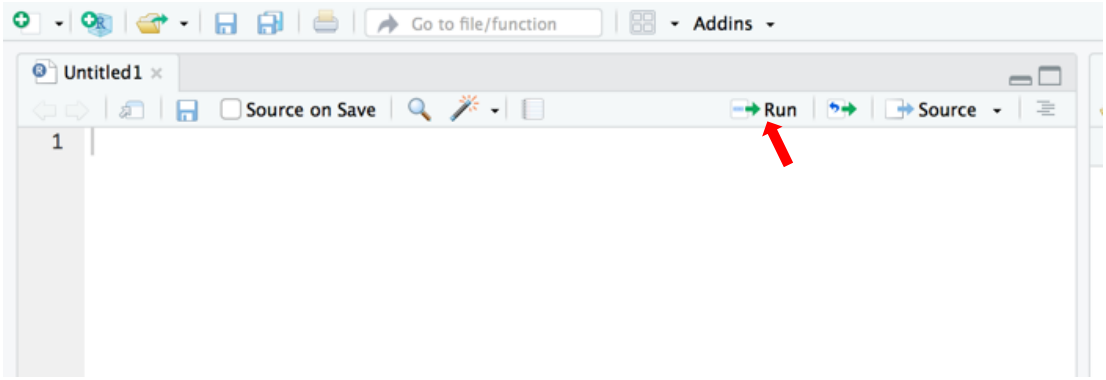
La ventana de Console ahora se ha situado debajo de la ventana de Scripts **obteniendo esta configuración nueva de cuatro ventanas**:



Ventana de Scripts o de edición (superior izquierda).

Como hemos dicho, aquí trabajaremos con scripts. El archivo que tengamos abierto, en este caso Untitled.R lo podemos grabar con el nombre que queramos, fijándonos que es el script en cuestión el que tenemos abierto (ya que se pueden tener más de un script abiertos en diferentes pestañaas) y accediendo a *File, Save File as* y guardándolo como *nombre.R*.

En esta ventana de scripts se pueden escribir ordenes, comandos y ejecutarlos clickando en el icono de Run. Si nos ponemos a la derecha de la orden, se ejecuta la orden de esa línea. Si seleccionamos varias órdenes a la vez, y damos a **Run,** se ejecutan todas a la vez. Nos fijamos que cuando ejecutamos órdenes de desde el script, éstas aparecen en la pantalla inferior izquierda Console.
Es interesante perder un poco de tiempo editando los script



Ventana Console (Consola) (inferior izquierda)

Los comandos se pueden introducir aquí o en la ventana del Script. En principio para ver que el comando está funcionando y que vamos haciendo las cosas bien se puede trabajar aquí y luego ir pasando comandos y comentarios al script para dejar un código final trabajado y ordenado. Una forma de pasar comandos de la Console al Script sería desde la ventana superior derecha, clickar en la pestaña History, seleccionar los comandos que queramos que se han quedado en el Historial y clickar en "To Source", para llevarlos al Script. Igualmente se pueden enviar comandos u órdenes del historial a la Consola, seleccionando la órden u órdenes y clickando en "To Console".

En la ventana Console por tanto se pueden dar ordenes, ejecutarlas y ver los resultados, a no ser que sea un gráfico que aparecerá en la sección Plot de la ventana inferior derecha.

En la consola, aparece un ***prompt*** (>) que indica la línea de comandos del área de trabajo para comenzar con la sesión.

Tras introducir las instrucciones requeridas debe pulsarse la tecla ***Enter*** para realizar la ejecución de la línea.

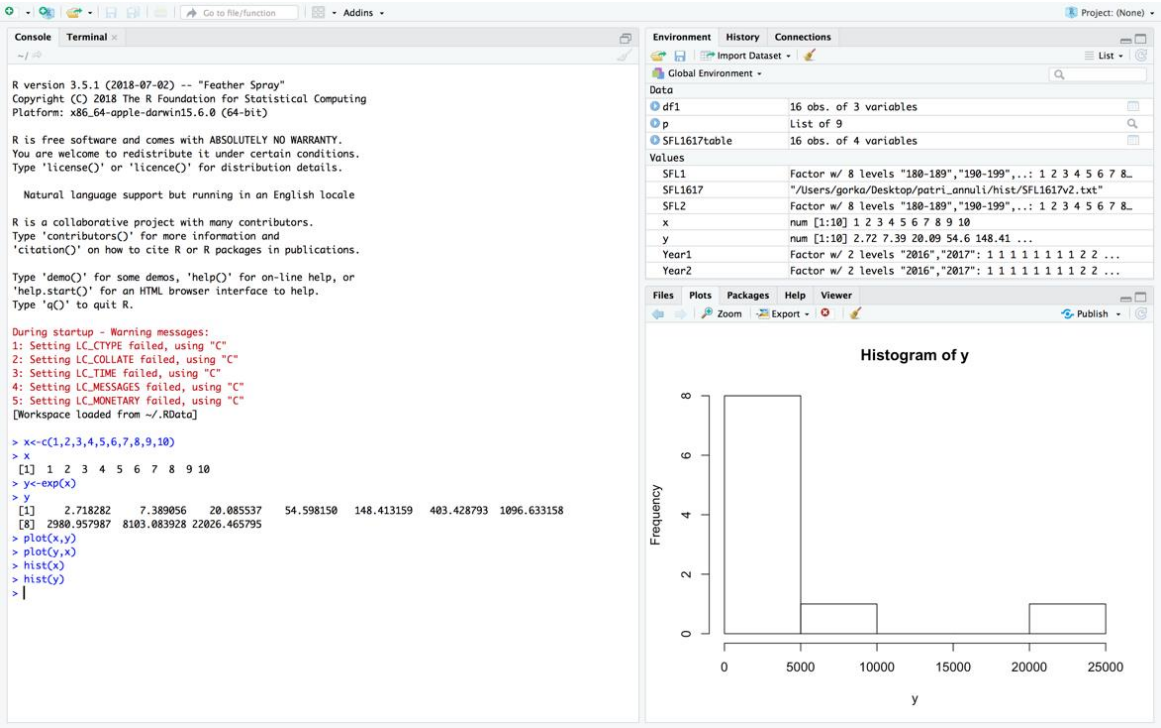
Nótese que el programa distingue entre mayúsculas y minúsculas.

Ventana Workspace (Environment-History) (superior derecha).

Aparecen todo el historial de comandos que he trabajado (pestaña History) y los objetos (variables, parámetros) creados (pestaña Environment). Desde la pestaña ENvironment puedo importar archivos txt (Import dataset ->From Text (base)) o excel (Import dataset->From Excel…)

Ventana Files-Plots-Packages-Help (inferior derecha).

Desde aquí veo los gráficos (Pestaña Plots), cargo paquetes o librerías para poder trabajar ciertos aspectos en R (Pestaña Pacakges y buscar y clickar el paquete quiequiero) o instalo paquetes (Pestaña Packages -> INstall) si no estñan instalados y los cargo después de instalarlos.



Si necesitamos ayuda podemos ir directamente a la sección de ayuda (Help) en este mismo Panel. En el menú de opciones para resolver cuestiones sobre R o Rstudio y en la pestaña Help de la ventana inferior derecha para ayuda específica de R. Otra forma muy rápida consiste es escribir, por ejemplo, ***?plot()*** en la Consola y nos saca la ayuda de la función plot en la ventana inferior derecha. Otro comando útil es Ctrl + L, para limpia la consola.

Se detallan, además, unas instrucciones básicas:

- help(): sistema on-line de ayuda
- help.start(): sistema de ayuda HTML
- q(): salir de *R*

La introducción de **comentarios** se realiza anteponiendo el símbolo **#** de forma que todo el texto que sigue al símbolo es tratado como un comentario.

Cuando se sale de *R* se ofrece la posibilidad de guardar la sesión de forma que pueda continuarse el trabajo en diferentes sesiones.

Al iniciar una sesión, por defecto, el programa busca los archivos solicitados en un determinado directorio. En ese mismo directorio guarda los archivos generados durante la sesión.

El directorio en el que se trabaja puede obtenerse con la función **getwd()**.

Es aconsejable utilizar distintos directorios para diferentes proyectos

EJERCICIOS DE CLASE

Para hacernos una idea sobre cómo funciona *R* veamos un ejemplo de una sesión.

A continuación comenzamos propiamente con la sesión de *R.* Tras teclear en la Consola el texto tras el prompt > se debe pulsar Enter. Por ejemplo, creamos en la consola un vector:

Creamos el vector **x** concatenando mediante el argumento **c** 20 elementos

```
x <- c(1.2,2.7,3.2,4.5,3.3,4.4,5.6,7,9,2.3,5,4.3,8.1,3.4,5.1,6.8,1.9,1.8,3,2.5)
```

observamos el output (los elementos del vector creado), escribiendo el nombre del objeto (x) y dandole a ENTER

```
x
```

podemos ordenar los datos

```
sort(x)
```

Vemos que los comandos u órdenes que ejecutamos en la Consola aparecen en el panel History de la ventana superior derecha. Como hemos visto antes, podemos pasar ese comando del panel History al script si queremos ejecutarlo desde y guardarlo en el script. Lo hacemos seleccionando el comando en cuestión en el panel **History** y clickando **To Source**.

Desde el script lo podemos ejecutar mediante **RUN** (arriba/derecha en el panel del script). Todo lo que pasemos al script o escribamos directamente en el script podemos añadirle un comentario (mediante #) para definir lo que hace cada orden y así acordarnos cuando abramos otra vez el script.

En las líneas anteriores se observa que, después de introducir las instrucciones y tras pulsar Enter en la consola o RUN en el script, el programa nos va devolviendo las salidas correspondientes a cada sentencia en la Consola. SI es un Plot la salida será en el panel de

Plots.

Al igual que en la consola, en el script, cuando creamos un objeto (ejemplo, objeto **x**) no proporciona directamente ninguna salida; sencillamente almacena el valor de x. Si queremos visualizar ese valor debemos escribir el objeto y ejecutar:

x

que nos devuelve los 20 valores de x que se habían almacenado en la sentencia anterior. Para empezar a trabajar con el programa realizaremos una de las tareas más sencillas que pueden ser llevadas a cabo en R; se trata de la introducción de una operación matemática simple y la obtención del resultado devuelto por el programa. Por ejemplo,

Miramos la longitud del vector (con length), es decir, el número de elementos

```
length(x)  # todo lo que se ponga detras del símbolo almoadilla no se ejecuta, es
interesante para poner comentarios respecto a las ordenes que queremos ejecutar
```

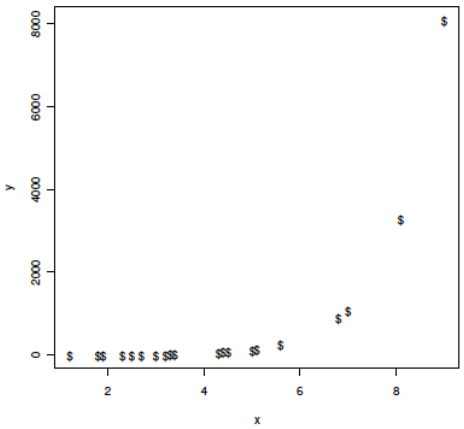
```
[1] 20
```

Función exponencial

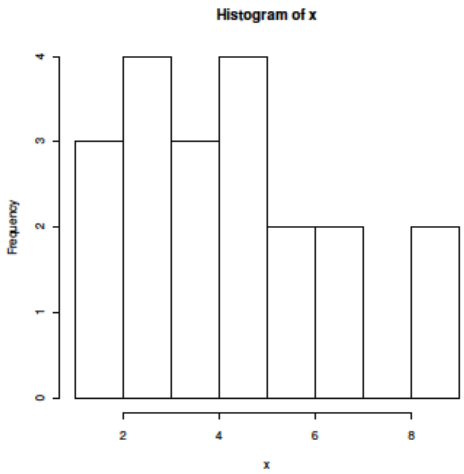
```
y<-exp(x). # función exponencial
y  # Presento los Valores de la función exponencial
```

```
[1] 3.320117 14.879732 24.532530 90.017131 27.112639  81.450869
[7] 270.426407 1096.633158 8103.083928 9.974182 148.413159  73.699794
[13] 3294.468075   29.964100  164.021907  897.847292 6.685894         6.049647
[19] 20.085537 12.182494
```

```
plot(x,y) # Gráfico la función
```




```
hist(x)  #Histograma de los valores x
```



Histograma nos dice que hay 3 datos entre 0 y 2, 4 datos entre 2 y 4, etc..

```
5*8
[1] 40

exp(1)
[1] 2.718282
```

Para asignar un valor a una variable se utilizan los símbolos <-. Para insertar comentarios basta anteponerles el símbolo #:

Definimos la variable x

```
x<-5
x

[1] 5

x^3
[1] 125
```

Cuando falta algún símbolo para completar una sentencia aparece el símbolo+:

```
sqrt(3)
```

INTRODUCCIÓN DE DATOS

Para construir un vector se utiliza la sentencia c():

```
pesoenkg<-c(5,2.8,3.7,4.6,8.1,3.2)
pesoenkg
[1] 5.0 2.8 3.7 4.6 8.1 3.2
```

```
pesoengr<-pesoenkg*1000
pesoengr
[1] 5000 2800 3700 4600 8100 3200
```

En lugar de utilizar la construcción anterior se puede emplear la sentencia scan(). De este modo los datos se introducen de uno en uno mediante la tecla Enter. Para indicarle al programa que hemos terminado de introducir datos se debe pulsar dos veces esa tecla:

Para construir series de valores, por ejemplo los números impares comprendidos entre 1 y 65, utilizamos la función seq(). Mediante rep() es posible repetir un patrón dado, incluso caracteres:

```
seq(1,65,2)
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
[26] 51 53 55 57 59 61 63 65

#  repetir el valor 98 5 veces

rep(98,5)
[1] 98 98 98 98 98

rep(c("sí","no"),3)
[1] "si" "no" "si" "no" "si" "no"
```

Los elementos no numéricos se escriben entre comillas.

LECTURA DE DATOS DE UN ARCHIVO

Cuando trabajamos con datos de un experimento o estudio y queremos hacerles un tratamiento para hacer plots o análisis estadísticos, es muy común crear un archivo con una hoja de cálculo EXCEL y pasarlo a *R*.

Por ello, vamos a crear un archivo en excel como el de abajo. Es aconsejable que no haya espacios entre las palabras de una variable (se pueden poner puntos):

Costo.unit	Costo.mat	Costo.mano.de.obra
13.59	87	80
15.71	78	95
15.97	81	106
20.21	65	115
24.64	51	128

Creamos la tabla en Excel y la guardamos de dos formas: (1) como archivo **excel (xlsx)** (costes.xlsx, por ejemplo) y (2) como **txt - delimitado por tabulaciones** (costes.txt). En este último caso hay que desplegar el menú de extensiones de archivo dentro de las opciones de guardado para encontrar la extensión "txt delimiatado por tabulaciones" (crea un txt con las columnas delimitadas por tabulaciones).

Podemos **generar un nuevo directorio en la carpeta DOCUMENTOS**. Por ejemplo, **Practica1** y guardar los archivos aquí.

Pinchando con el botón derecho en los archivos guardados y clickando en propiedades nos dice que los archivos están en la ruta o path (saber la ruta es importante para después):

C:/Users/Docencia/Documents/Practica1

Ahora vemos donde está trabajando R, escribiendo:

```
getwd()
```

Seguramente está trabajando en otro directorio. Es interesante trabajar en R en el mismo directorio donde tenemos los archivos que vamos a importar Por tanto, si no estamos en el directorio donde hemos guardado los archivos excel y txt, podemos especificar el nuevo directorio de trabajo en R (el que tengamos los archivos):

```
setwd("C:/Users/Docencia/Documents/Practica1") # situamos R en este directorio

getwd()  # comprobamos que estamos en el Nuevo directorio de trabajo
```

Lectura del archivo.txt con comandos

Una vez situado R en el directorio donde tenemos los archivos .txt y .excel, ahora, primero queremos leer (importar) el archivo de texto (txt) **costes.txt** en R. Lo hacemos con el comando **read.table.** Creamos un nuevo objeto, por ejemplo "costesdatatxt" para importar con ese nombre el archivo txt.

```
costesdatatxt <-read.table(file="costes.txt",header=TRUE,dec=".",sep="\t")
```

El argumento header=TRUE o header=T significa que la primera línea del marco de datos contiene los nombres de las variables.

El argumento dec="." significa que lea como decimal los puntos (hay que fijarse en el txt como está guardado).

El argumento sep="\t" significa que considera que cada columna de datos (para cada variable) está separada mediante un espacio con tabulación, que es como hemos grabado el archivo costes.txt desde excel.

Estos argumentos por tanto cambian ".", "," dependiendo cómo son los decimales y como es el espacio de separación de columnas/variables.

Como anteriormente se ha especificado el directorio de trabajo con setwd (directorio donde tenemos los archivos excel y txt generados) en **file=** solo es necesario poner el nombre del archivo "costes.txt".

Si no hubiésemos especificado el directorio de trabajo en **file=** tendríamos que haber especificado la **ruta** donde se encuentra el archivo **costes.txt**, acabando con el archivo que queremos leer, tal que así:

```
costesdatatxt<-read.table(file="C:/Users/Docencia/Documents/Practica1/costes.txt",
header=TRUE,dec=".", sep="\t")
```

ahora, escribiendo **costes*datatxt*** *(el objeto que hemos creado para importar el archivo)* nos salen los datos que hemos importado:

costesdatatxt

ID		Costo.unit	Costo.mat	Costo.mano.de.obra
1	1	13.59	87	80
2	2	15.71	78	95
3	3	15.97	81	106

Podemos darle también a

```
View(costedatatxt)
```

Si queremos calcular la media del coste unitario (**Costo.unit**) tenemos que trabajar con la variable Costo.unit. Para ello no es suficiente con **mean(Costo.unit)**, daría error.

Para trabajar con y analizar variables de un archivo de datos importado, primero tendremos que hacer accesible en R el objeto que contenga estas variables. Esto se hace mediante el comando **attach**. Con este comando se crea un **marco de datos o data frame** accesible.

Introducimos el comando attach, así:

```
attach(costesdatatxt)
```

Pedimos que nos enseñe (con el comando **names**) los nombres de las variables de ese marco de datos:

names(costesdatatxt)

el output es el siguiente:

[2] "Costo.unit" "Costo.mat" "Costo.mano.de.obra

ahora llamamos a una variable para ver si el marco de datos (data frame) lo ha pegado en R bien y en ese caso podemos trabajar con las variables:

Costo.mat
[1] 87 78 81 65 51

Podemos calcular la media de los datos de una variable:

Mean(Costo.mat)

Parece que esta todo OK y que el data frame de los datos importado esta en R (hemos hecho bien el attach)

Simbolo $ en vez de attach para acceder a las variables y trabajar con ellas:

Otro modo de acceder a una variable, por ejemplo a "Costo.mat" del marco de datos **costesdatatxt**, consiste en seleccionar la correspondiente columna de la variable de esta manera. Se pone en primer lugar el nombre del objeto y despus del símbolo $ se pone el nombre de la variable que queremos trabajar, así:

costesdatatxt$Costo.mat.

Si queremos conocer de qué tipo es una variable concreta hacemos lo siguiente:

class(Costo.mat)

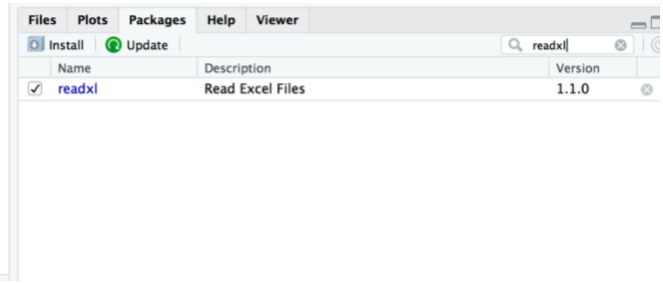
[1] “integer”

Lo que significa que Costo.mat es un vector con datos de tipo numérico y de números enteros.

Lectura del archivo.xlsx con comandos

Para trabajar con archivos excel necesitamos instalar/activar ciertos paquetes. En el **panel** donde se sitúan los paquetes (**Package, panel inferior derecha)** buscamos y activamos el

paquete **readxl** clickando (visto) a la izquierda del paquete. Hacemos lo mismo con el paquete **openxlsx**.



Si lss buscamos y no lo encuentra es que no está instalado; los instalamos con **Install** en el mismo panel**. Install** -> Buscar-> Seleccionar-> Instalar

Una vez instalados hay que activarlos clickando en ellos en la zona donde se ha comentado anteriormente, o se puede también activar de este modo llamando a la librería o paquete con el comando **library**:

```
library(readxl)

library(openxlsx)
```

Una vez activado el paquete, leemos el fichero .xlsx asignándole otro objeto mediante:

```
costesdataxlsx<-read_excel(file=" costes.xlsx")
```

O mediante:

```
costesdataxlsx<-read.xlsx("costes.xlsx")

costesdataxlsx  # visualizamos el data frame o marco de datos
```

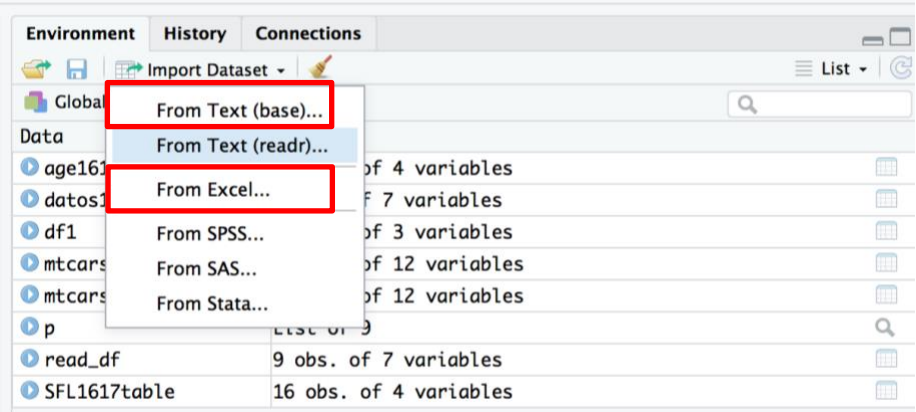
	ID	Costo.unit	Costo.mat	Costo.mano.de.obra
1	1	3.59	87	80
2	2	15.71	78	95
3	3	15.97	81	106

Pero más fácil....

se pueden IMPORTAR archivos .txt y .xlsx DESDE EL PANEL ENVIRONMENT

La otra forma sería, accediendo a importar el archivo xlsx o txt desde la ventana superior derecha, en la pestaña Environment. Concretamente, desde la pestaña **Import dataset**,

seleccionando “**From Text(base)”** para txt y “**From Excel**” para archivos excel.



Buscamos en el Browser el archivo deseado y lo importamos. En el caso de que sea txt hay que seleccionar tab como delimiter.

FUNCIÓN data.frame

Mediante la función **data.frame** podemos crear un marco de datos. Por ejemplo, creamos tres objetos (vectores), el primero una lista de números, el segundo esos números al cuadrado y el tercero esos números al cubo:

```
num<-c(1,2,3,4,5,6,7)
numcua<-num^2
numcub<-num^3
```

Luego juntamos los objetos en un marco de datos o data frame, donde se disponen esos objetos o vectores en columnas:

```
marco.datos<-data.frame(num,numcua,numcub)
```

Ver marco datos:

```
marco.datos
```

Se ve algo asi:

num	numcua	numcub
1	1	1
2	2	4	8
3	3	9	27
4	4	16	64
5	5	25	125

6 6 36 216
7 7 49 343

En este caso NO ES NECESARIO ejecutar la orden **attach** ya que la base de datos (data frame) la hemos generado en R.

Para ver los elementos de la variable o vector num, directamnte hago:

num

[1] 1 2 3 4 5 6 7

Calcular la media de numcua

mean(numcua)

[1] 20

SELECCIÓN DE ELEMENTOS DE UN OBJETO

Existe una función muy interesante en R, str(), que permite conocer cuál es la estructura de un determinado objeto que haya sido creado en una sesión. Por ejemplo, veamos qué nos dice esta función sobre el objeto A:

```
> str(marco.datos)
'data.frame': 7 obs. of 3 variables
$ num : num 1 2 3 4 5 6 7
$ numcua: num 1 4 9 16 25 36 49
$ numcub: num 1 8 27 64 125 216 343
```

La salida anterior nos indica que el objeto **marco.datos** es un marco de datos formado por 7 observaciones de 3 variables, denominadas *num, numcua y numcub*. Si queremos escoger la primera de esas variables podemos hacer

```
> marco.datos$num
[1] 1 2 3 4 5 6 7
```

De todos modos, como las variables habían sido creadas previamente, y ya eran accesibles, se pueden llamar directamente sin el símbnolo $.

Vemos los elementos de la segunda variable en primer lugar.

numcua

[1] 1 4 9 16 25 36 49

numcua[5]
[1] 25

Podemos así mismo efectuar selecciones condicionales, como por ejemplo:

numcua[numcua<4] # nos da los datos menores que el 4:

[1] 1

numcub[numcub==7] # nos da el dato que es igual a 7
[1] numeric 0

Para ver qué valores cumplen una cierta condición y conocer las posiciones correspondientes podemos utilizar la función which():

which(numcua>10) # en que posición están los datos mayores de 10 en numcua
[1] 4 5 6 7

which(numcub>0) # en que posición están los datos mayores de 0 en numcub

[1] 1 2 3 4 5 6 7

which(numcub>10) # en que posición están los datos mayores de 10 en numcub
[1] 3 4 5 6 7

which(numcub==7) # la posición del dato igual a 7

length(which(numcub>7) # número de datos mayores que 7 en numcub, es decir, longitud del vector que se compone con los datos de numcub que son mayores que 7.

Para eliminar valores de una variable hacemos:

numcua[-c(1,3,5,7)] # elimina los datos de las posiciones 1,3,5,7 y me presenta los datos que quedan despues de esa eliminación de datos

[1] 4 16 36

En ocasiones suele ser interesante saber qué elementos de un cierto vector b están en otro vector a:

a<-1:10
a

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

b<-c(1,3,7,15,20)
b

[1] 1 3 7 15 20

b[b%in%a] # los elementos de b que están en a

[1] 1 3 7
```

FUNCIONES

Para definir nuevas funciones en *R*, como por ejemplo f(x) = $2x^2+1$, se hace:

```
f1<-function(x) 2*x^2+1

f1(-5)  # nos da el valor de la función para x=-5

[1] 51
```

Ejemplo. Dada la función f(x) = sen(e^x)

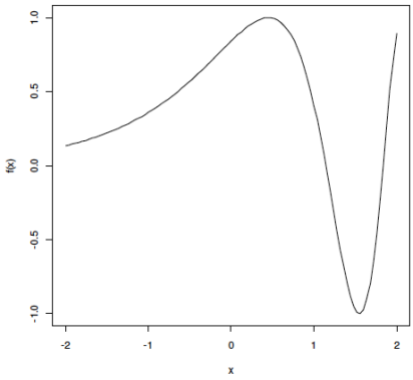
1º) Obtener f(-1), f(0) y f(1).
2º) Representación gráfica en el intervalo (-2,2).
3º) Calcular el área bajo la curva entre las abscisas x=0 y x=1.

```
f2<-function(x) sin(exp(x))

f2(-1);f2(0);f2(1)

[1] 0.3596376
[1] 0.841471
[1] 0.4107813

plot(f2,-2,2)
```



integrate(f2,0,1)
0.8749572 with absolute error < 9.7e-15

CÓMO GESTIONAR UNA SESIÓN DE *R*

Cuando se acaba una sesión de R, aparte de guardar el Script con el nombre_archivo.R, es posible guardar el espacio de trabajo (Workspace) con sus objetos, variables, etc. y el historial entero. Para ello:

Acceder a menu Tools, Global Options, RGeneral → clikar