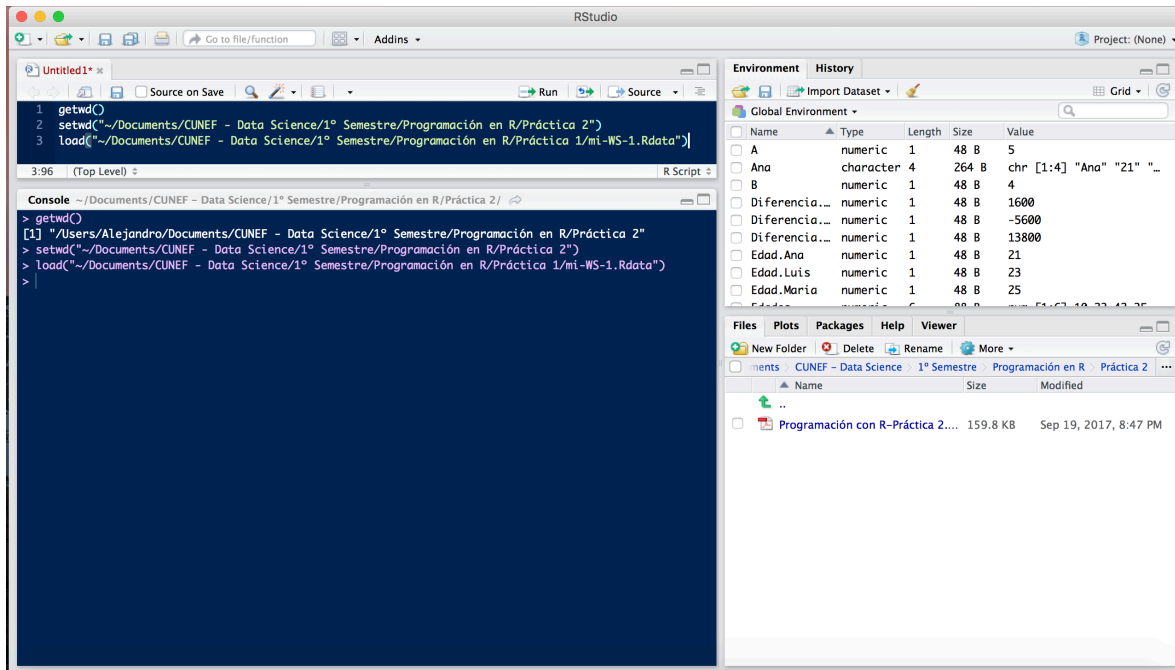


## Ejercicio 1. RECUPERAR WORKSPACE

Ejecutar `getwd()` para conocer dónde está el workspace del sistema y comprobado que se recupera el almacenado en la práctica 1. Utilizar `setwd("ruta de la carpeta")` para asignar la carpeta donde se desea que se guarde el workspace que se va a crear en la sesión o `load("ruta de la carpeta")` para cargar la que se desee en cada momento. Es recomendable una carpeta por cada práctica o trabajo, para tener diferenciados las tareas realizadas. Se puede utilizar el sistema de menús de R y de RStudio para este cometido.



## Ejercicio 2. ESTRUCTURAS DE DATOS

Creación de Data Frames. Un data.frame es una estructura donde almacenar objetos de diferentes tipos. En este ejercicio queremos crear una estructura por cada una de las personas de la tabla del ejercicio 2 de la práctica 1. Tras realizar la asociación, comprobar el resultado visualizando el contenido de workspace.

Para ello, creamos un data.frame con la sentencia data.frame(lista de objetos separada por comas). Análogamente lo podremos hacer con data.table o transformar directamente dt<-data.table(objeto tipo data.frame), de esta forma el nuevo objeto dt tendrá habilitadas todas las funciones de data.table, comprobado con str(dt).

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R script for creating the data frames.

```
1 Ana_data.frame <- data.frame(Edad.Ana, Gastos.Ana, Ingresos.Ana)
2 Luis_data.frame <- data.frame(Edad.Luis, Gastos.Luis, Ingresos.Luis)
3 Maria_data.frame <- data.frame(Edad.Maria, Gastos.Maria, Ingresos.Maria)
4 print(Ana_data.frame)
5 print(Luis_data.frame)
6 print(Maria_data.frame)
```
- Console:** Shows the output of the print statements.

```
> Ana_data.frame <- data.frame(Edad.Ana, Gastos.Ana, Ingresos.Ana)
> Luis_data.frame <- data.frame(Edad.Luis, Gastos.Luis, Ingresos.Luis)
> Maria_data.frame <- data.frame(Edad.Maria, Gastos.Maria, Ingresos.Maria)
> print(Ana_data.frame)
  Edad.Ana Gastos.Ana Ingresos.Ana
1       21    23400      25000
> print(Luis_data.frame)
  Edad.Luis Gastos.Luis Ingresos.Luis
1       23    32000      26400
> print(Maria_data.frame)
  Edad.Maria Gastos.Maria Ingresos.Maria
1       25    17400      31200
> |
```
- Environment Pane:** Displays the objects in the workspace.

Name	Type	Length	Size	Value
Maria	character	4	264 B	chr [1:4] "Maria" "25" "31200"
Nombre.Ana	character	1	96 B	"Ana"
Nombre.Luis	character	1	96 B	"Luis"
Nombre.Maria	character	1	96 B	"Maria"
Ana_data.frame	data.frame	3	944 B	1 obs. of 3 variables
Luis_data.frame	data.frame	3	944 B	1 obs. of 3 variables
Maria_data.frame	data.frame	3	944 B	1 obs. of 3 variables
s.Edades	logical	6	72 B	logi [1:6] TRUE FALSE TRUE T...
A	numeric	1	48 B	5
a	numeric	1	48 B	4
- Files Pane:** Shows the system library with various packages like abind, acepack, adabag, assertthat, backports, base64enc, BH, bindr, bindrcpp, bitops, boot, car, and caret.

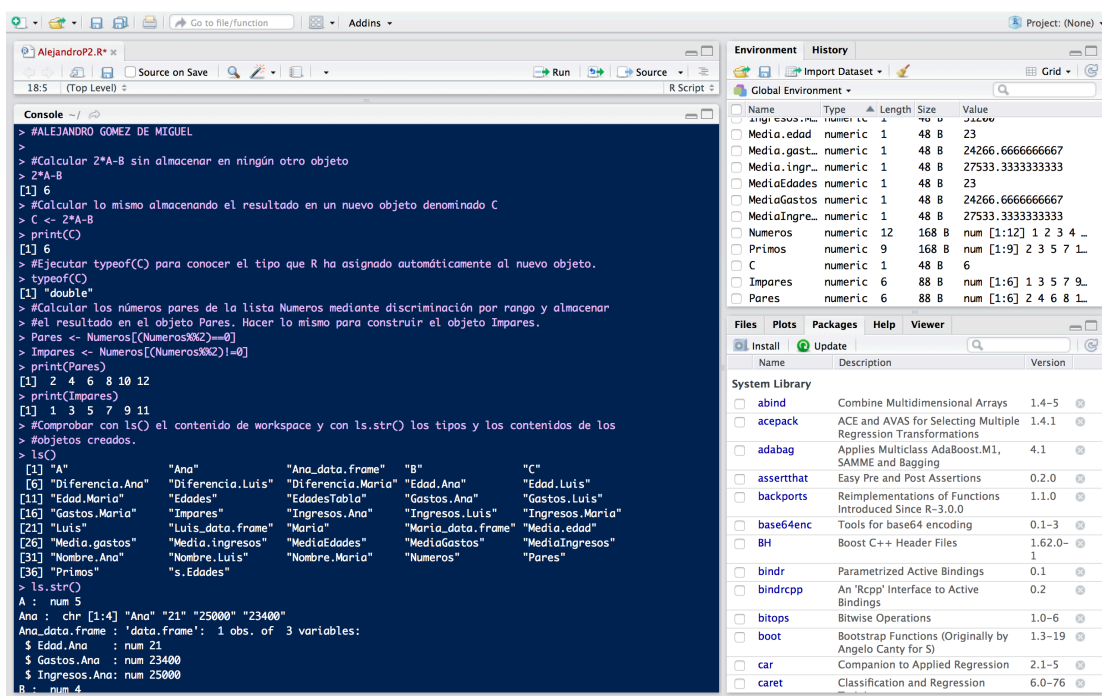
## Ejercicio 3. CREACIÓN DE SCRIPTS

Crear un script mediante el sistema de menus de R llamado “minombreP2.R”.

Dentro del scrip, escribid los comentarios tras el símbolo # que anula el resto de la línea del proceso de ejecución. Añadid un título con vuestro nombre completo.

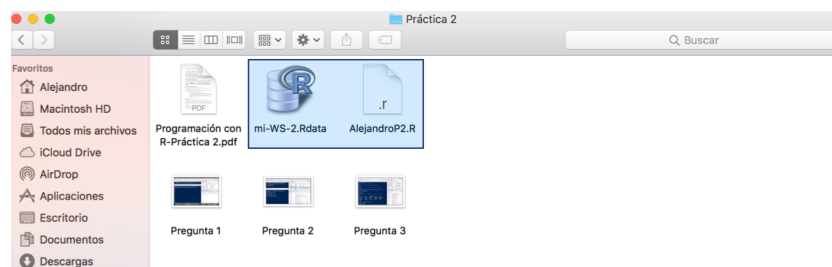
En cada línea del script debéis escribir una sentencia que posteriormente podrá ser ejecutada cuantas veces se desee. Se pide escribir líneas para ejecutar las siguientes sentencias:

- Calcular  $2*A-B$  sin almacenar en ningún otro objeto
- Calcular lo mismo almacenando el resultado en un nuevo objeto denominado C
- Ejecutar `typeof(C)` para conocer el tipo que R ha asignado automáticamente al nuevo objeto.
- Calcular los números pares de la lista Numeros mediante discriminación por rango y almacenar el resultado en el objeto Pares. Hacer lo mismo para construir el objeto Impares.
- Comprobar con `ls()` el contenido de workspace y con `ls.str()` los tipos y los contenidos de los objetos creados.
- Salvar el workspace y el script. Cerrar R.



```
#ALEJANDRO GOMEZ DE MIGUEL
>
> #Calcular 2*A-B sin almacenar en ningún otro objeto
> 2*A-B
[1] 6
> #Calcular lo mismo almacenando el resultado en un nuevo objeto denominado C
> C <- 2*A-B
> print(C)
[1] 6
> #Ejecutar typeof(C) para conocer el tipo que R ha asignado automáticamente al nuevo objeto.
> typeof(C)
[1] "double"
> #Calcular los números pares de la lista Numeros mediante discriminación por rango y almacenar
> #el resultado en el objeto Pares. Hacer lo mismo para construir el objeto Impares.
> Pares <- Numeros[(Numeros%%2)==0]
> Impares <- Numeros[(Numeros%%2)!=0]
> print(Pares)
[1] 2 4 6 8 10 12
> print(Impares)
[1] 1 3 5 7 9 11
> #Comprobar con ls() el contenido de workspace y con ls.str() los tipos y los contenidos de los
> #objetos creados.
> ls()
[1] "A" "Ana" "Ana_data.frame" "B" "C"
[6] "Diferencia.Ana" "Diferencia.Luis" "Diferencia.Maria" "Edad.Ana" "Edad.Luis"
[11] "Edad.Maria" "Edades" "EdadesTabla" "Gastos.Ana" "Gastos.Luis"
[16] "Gastos.Maria" "Impares" "Ingresos.Ana" "Ingresos.Luis" "Ingresos.Maria"
[21] "Luis" "Luis_data.frame" "Maria" "Maria_data.frame" "Media.edad"
[26] "Media.gastos" "Media.ingresos" "MediaEdades" "MediaGastos" "MediaIngresos"
[31] "Nombre.Ana" "Nombre.Luis" "Nombre.Maria" "Numeros" "Pares"
[36] "Primos" "s.Edades"
> ls.str()
A : num 5
Ana : chr [1:4] "Ana" "21" "25000" "23400"
Ana_data.frame : 'data.frame': 1 obs. of 3 variables:
 $ Edad.Ana : num 21
 $ Gastos.Ana : num 23400
 $ Ingresos.Ana : num 25000
B : num 4
```

Name	Type	Length	Size	Value
Ingresos.Luis	numeric	12	70 B	25000
Media.edad	numeric	1	48 B	23
Media.gast...	numeric	1	48 B	24266.6666666667
Media.ingr...	numeric	1	48 B	27533.3333333333
MediaEdades	numeric	1	48 B	23
MediaGastos	numeric	1	48 B	24266.6666666667
MediaIngre...	numeric	1	48 B	27533.3333333333
Numeros	numeric	12	168 B	num [1:12] 1 2 3 4 ...
Primos	numeric	9	168 B	num [1:9] 2 3 5 7 1...
C	numeric	1	48 B	6
Impares	numeric	6	88 B	num [1:6] 1 3 5 7 9...
Pares	numeric	6	88 B	num [1:6] 2 4 6 8 1...



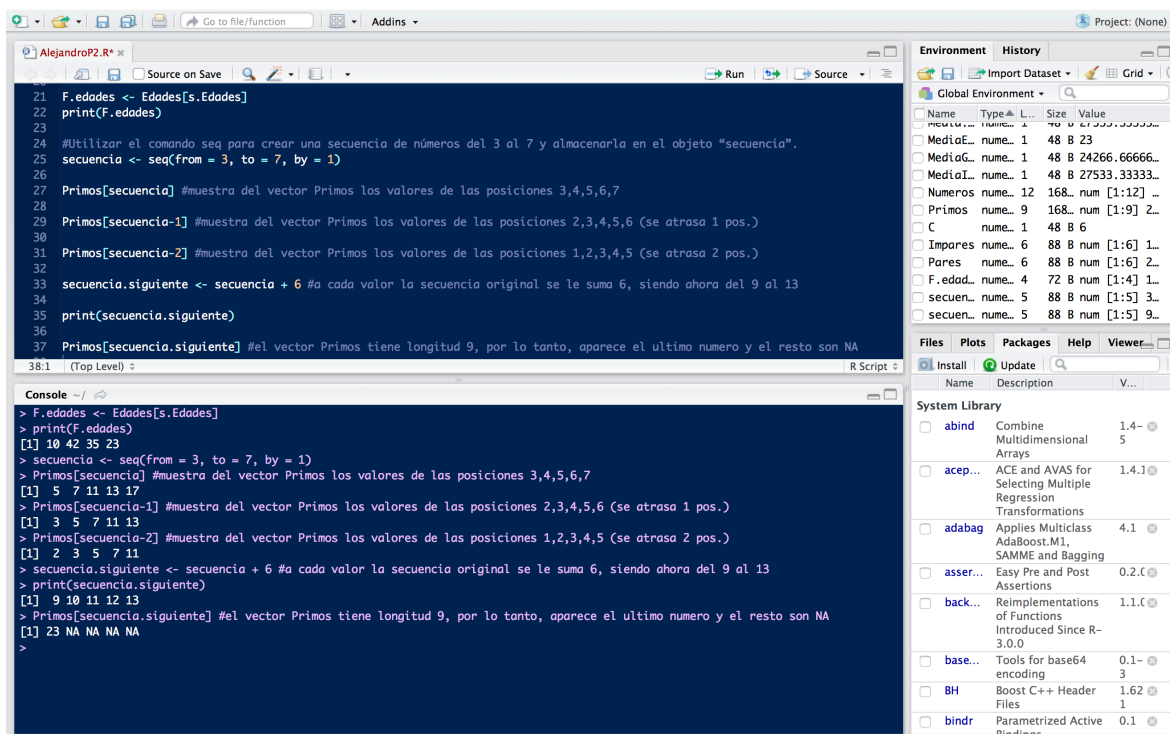
## Ejercicio 4.

Abrir R y recuperar el trabajo del ejercicio anterior. Comprobad el contenido de las variables.

Teniendo en cuenta que la variable `s.edades` representa el sexo (femenino = True) de la persona a la que corresponde la edad del vector `Edades`, se desea crear un objeto `F.edades` donde almacenar las edades de las personas de sexo femenino. Añadir el comando al script de trabajo. Utilizar el comando `print()` para comprobar el resultado. Utilizar el comando `seq` para crear una secuencia de números del 3 al 7 y almacenarla en el objeto "secuencia".

Ejecutar cada una de las sentencias siguientes y comentar el resultado:

```
>Primos[secuencia]; Primos[secuencia-1]; Primos[secuencia-2]; secuencia.siguiente<-secuencia+6;  
secuencia.siguiente; Primos[secuencia.siguiente]
```



The screenshot shows the RStudio interface with a script editor and a console. The script editor contains the following R code:

```
21 F.edades <- Edades[s.Edades]
22 print(F.edades)
23
24 #Utilizar el comando seq para crear una secuencia de números del 3 al 7 y almacenarla en el objeto "secuencia".
25 secuencia <- seq(from = 3, to = 7, by = 1)
26
27 Primos[secuencia] #muestra del vector Primos los valores de las posiciones 3,4,5,6,7
28
29 Primos[secuencia-1] #muestra del vector Primos los valores de las posiciones 2,3,4,5,6 (se atrasa 1 pos.)
30
31 Primos[secuencia-2] #muestra del vector Primos los valores de las posiciones 1,2,3,4,5 (se atrasa 2 pos.)
32
33 secuencia.siguiente <- secuencia + 6 #a cada valor la secuencia original se le suma 6, siendo ahora del 9 al 13
34
35 print(secuencia.siguiente)
36
37 Primos[secuencia.siguiente] #el vector Primos tiene longitud 9, por lo tanto, aparece el ultimo numero y el resto son NA
```

The console shows the output of the executed commands:

```
> F.edades <- Edades[s.Edades]
> print(F.edades)
[1] 10 42 35 23
> secuencia <- seq(from = 3, to = 7, by = 1)
> Primos[secuencia] #muestra del vector Primos los valores de las posiciones 3,4,5,6,7
[1] 5 7 11 13 17
> Primos[secuencia-1] #muestra del vector Primos los valores de las posiciones 2,3,4,5,6 (se atrasa 1 pos.)
[1] 3 5 7 11 13
> Primos[secuencia-2] #muestra del vector Primos los valores de las posiciones 1,2,3,4,5 (se atrasa 2 pos.)
[1] 2 3 5 7 11
> secuencia.siguiente <- secuencia + 6 #a cada valor la secuencia original se le suma 6, siendo ahora del 9 al 13
> print(secuencia.siguiente)
[1] 9 10 11 12 13
> Primos[secuencia.siguiente] #el vector Primos tiene longitud 9, por lo tanto, aparece el ultimo numero y el resto son NA
[1] 23 NA NA NA NA
>
```