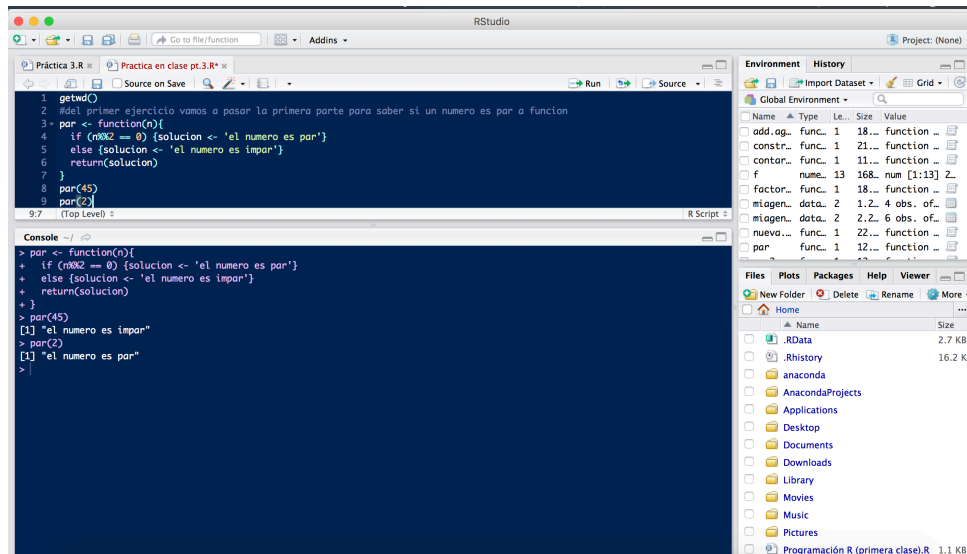


Ejercicio 1. Programación de Scripts y Funciones.

Ejercita los códigos siguientes para aprender a manejar las instrucciones de control de R. Construye uno o varios scripts para comprobar su funcionamiento. Transforma a funciones más generales.

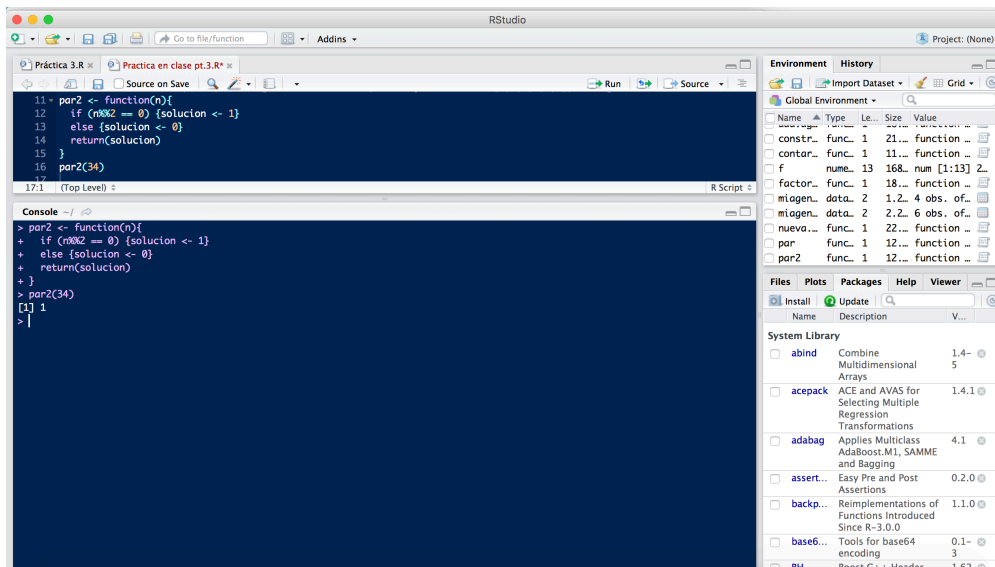
Función par:



```
1 getwd()
2 #del primer ejercicio vamos a pasar la primera parte para saber si un numero es par a funcion
3 par <- function(n){
4   if (n%%2 == 0) {solucion <- 'el numero es par'}
5   else {solucion <- 'el numero es impar'}
6   return(solucion)
7 }
8 par(45)
9 par(2)
```

```
> par <- function(n){
+   if (n%%2 == 0) {solucion <- 'el numero es par'}
+   else {solucion <- 'el numero es impar'}
+   return(solucion)
+ }
> par(45)
[1] "el numero es impar"
> par(2)
[1] "el numero es par"
> |
```

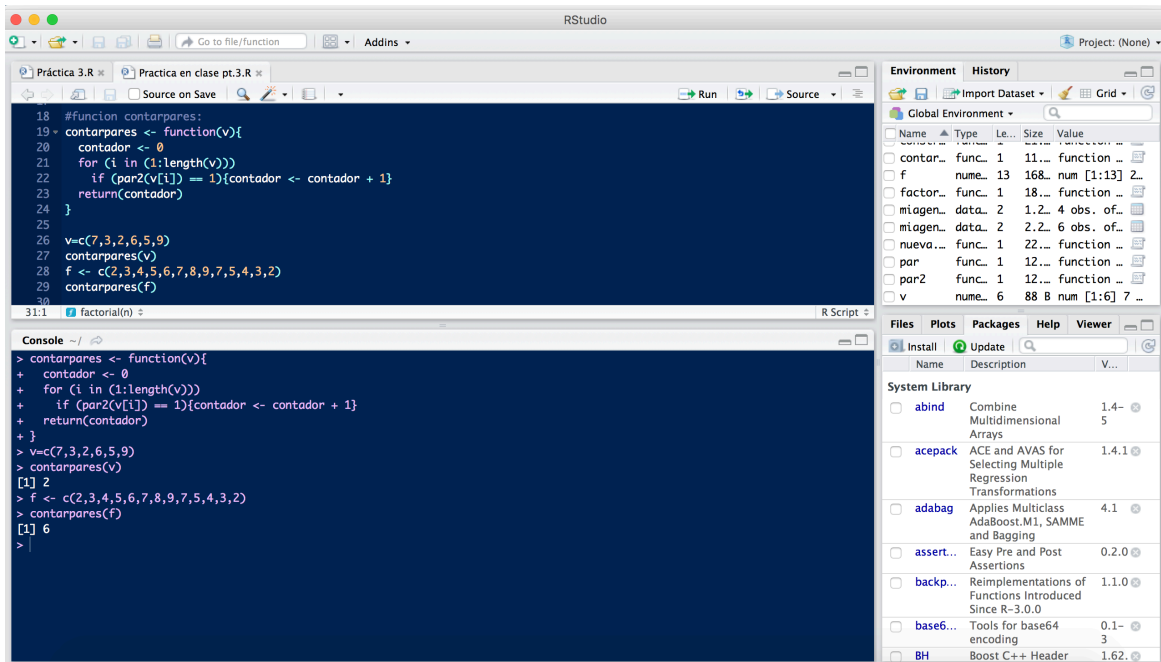
Función par2, versión booleana:



```
11 par2 <- function(n){
12   if (n%%2 == 0) {solucion <- 1}
13   else {solucion <- 0}
14   return(solucion)
15 }
16 par2(34)
```

```
> par2 <- function(n){
+   if (n%%2 == 0) {solucion <- 1}
+   else {solucion <- 0}
+   return(solucion)
+ }
> par2(34)
[1] 1
> |
```

Función contar pares, ¿cuántos números pares hay un vector v?



The screenshot shows the RStudio interface. The script editor contains the following code:

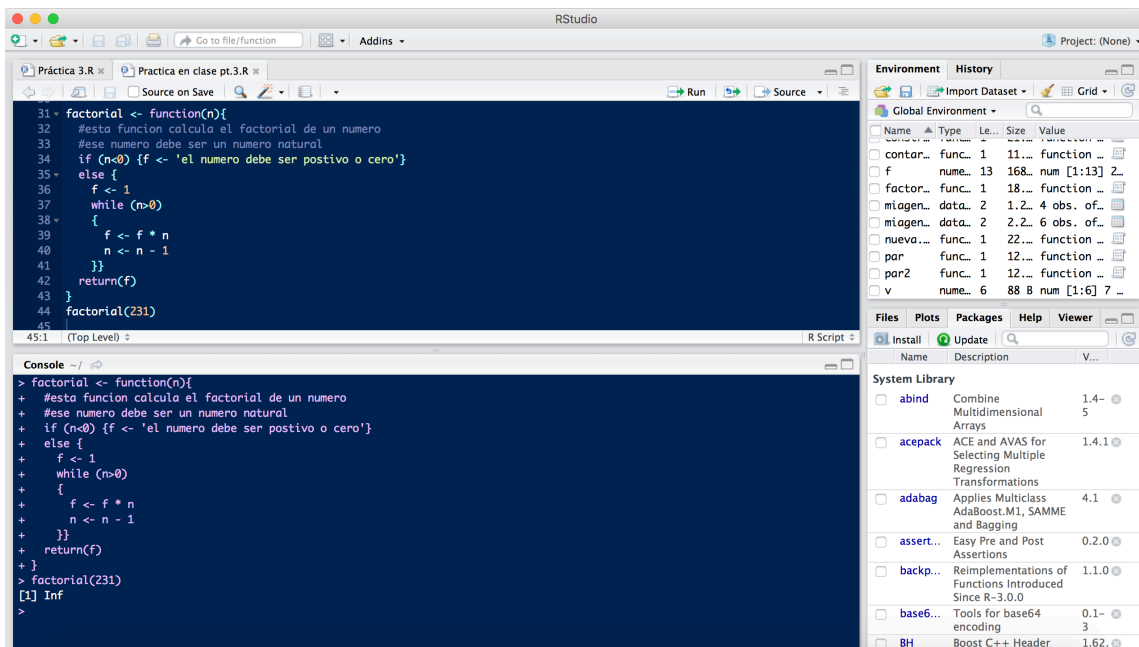
```
18 #funcion contar pares:
19 contar pares <- function(v){
20   contador <- 0
21   for (i in 1:length(v)){
22     if (par2(v[i]) == 1){contador <- contador + 1}
23   }
24   return(contador)
25 }
26 v=c(7,3,2,6,5,9)
27 contar pares(v)
28 f <- c(2,3,4,5,6,7,8,9,7,5,4,3,2)
29 contar pares(f)
30
```

The console shows the execution of the function:

```
> contar pares <- function(v){
+   contador <- 0
+   for (i in 1:length(v)){
+     if (par2(v[i]) == 1){contador <- contador + 1}
+   }
+   return(contador)
+ }
> v=c(7,3,2,6,5,9)
> contar pares(v)
[1] 2
> f <- c(2,3,4,5,6,7,8,9,7,5,4,3,2)
> contar pares(f)
[1] 6
>
```

The Environment pane on the right shows the global environment with variables: contar... (function), f (numeric vector), factor... (function), miagen... (data frame), miagen... (data frame), nueva... (function), par (function), par2 (function), and v (numeric vector).

Función factorial, calcula el factorial de un numero:



The screenshot shows the RStudio interface. The script editor contains the following code:

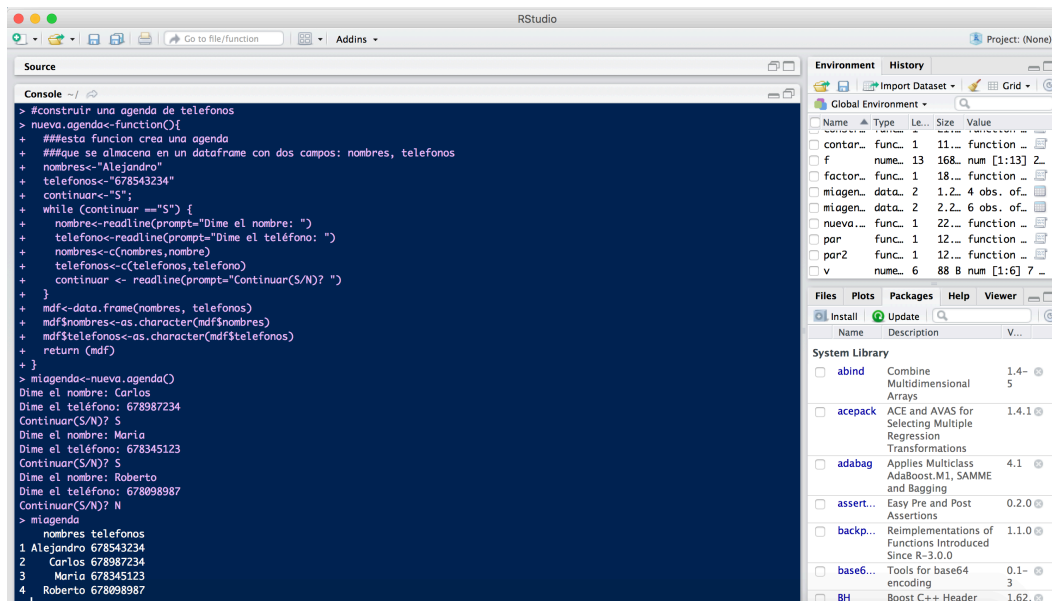
```
31 factorial <- function(n){
32   #esta funcion calcula el factorial de un numero
33   #ese numero debe ser un numero natural
34   if (n<0) {f <- 'el numero debe ser positivo o cero'}
35   else {
36     f <- 1
37     while (n>0)
38     {
39       f <- f * n
40       n <- n - 1
41     }
42     return(f)
43   }
44   factorial(231)
45 }
```

The console shows the execution of the function:

```
> factorial <- function(n){
+   #esta funcion calcula el factorial de un numero
+   #ese numero debe ser un numero natural
+   if (n<0) {f <- 'el numero debe ser positivo o cero'}
+   else {
+     f <- 1
+     while (n>0)
+     {
+       f <- f * n
+       n <- n - 1
+     }
+     return(f)
+   }
+ }
> factorial(231)
[1] Inf
>
```

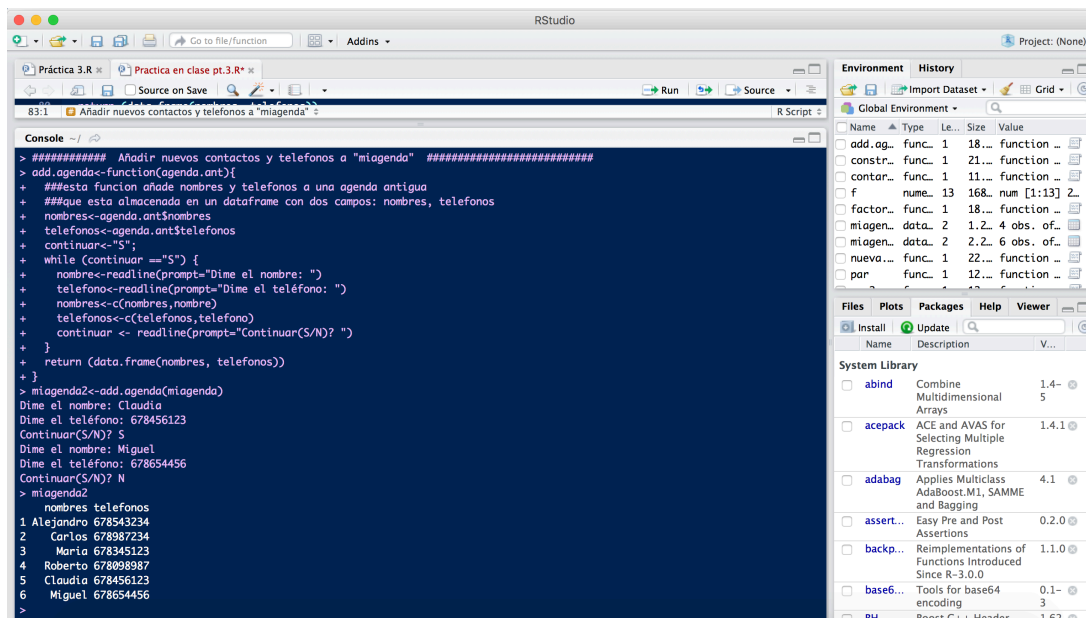
The Environment pane on the right shows the global environment with variables: contar... (function), f (numeric vector), factor... (function), miagen... (data frame), miagen... (data frame), nueva... (function), par (function), par2 (function), and v (numeric vector).

Crear una agenda de teléfonos, introducir 3 contactos:



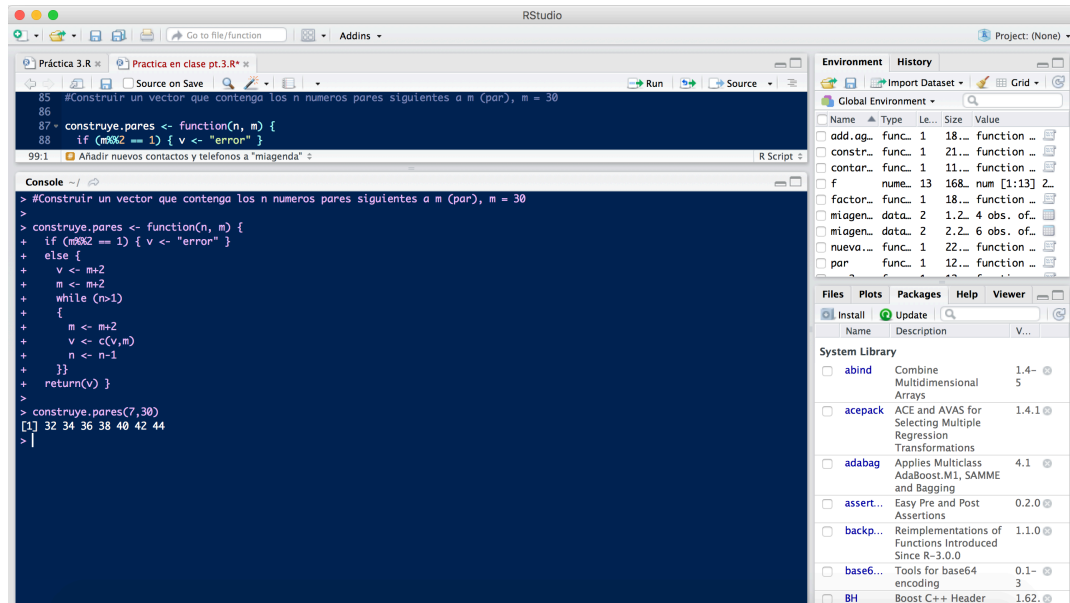
```
> #construir una agenda de telefonos
> nueva.agenda<-function(){
+   ##esta funcion crea una agenda
+   ##que se almacena en un dataframe con dos campos: nombres, telefonos
+   nombres<-"Alejandro"
+   telefonos<-"678543234"
+   continuar<-"S";
+   while (continuar == "S") {
+     nombre<-readline(prompt="Dime el nombre: ")
+     telefono<-readline(prompt="Dime el teléfono: ")
+     nombres<-c(nombres,nombre)
+     telefonos<-c(telefonos,telefono)
+     continuar <- readline(prompt="Continuar(S/N)? ")
+   }
+   mdf<-data.frame(nombres, telefonos)
+   mdf$nombres<-as.character(mdf$nombres)
+   mdf$telefonos<-as.character(mdf$telefonos)
+   return (mdf)
+ }
> miagenda<-nueva.agenda()
Dime el nombre: Carlos
Dime el teléfono: 678987234
Continuar(S/N)? S
Dime el nombre: Maria
Dime el teléfono: 678345123
Continuar(S/N)? S
Dime el nombre: Roberto
Dime el teléfono: 678098987
Continuar(S/N)? N
> miagenda
  nombres telefonos
1 Alejandro 678543234
2 Carlos 678987234
3 Maria 678345123
4 Roberto 678098987
```

Añadir a la agenda antigua 2 contactos más:



```
> ##### Añadir nuevos contactos y telefonos a "miagenda" #####
> add.agenda<-function(agenda,ant){
+   ##esta funcion añade nombres y telefonos a una agenda antigua
+   ##que esta almacenada en un dataframe con dos campos: nombres, telefonos
+   nombres<-agenda$ant$nombres
+   telefonos<-agenda$ant$telefonos
+   continuar<-"S";
+   while (continuar == "S") {
+     nombre<-readline(prompt="Dime el nombre: ")
+     telefono<-readline(prompt="Dime el teléfono: ")
+     nombres<-c(nombres,nombre)
+     telefonos<-c(telefonos,telefono)
+     continuar <- readline(prompt="Continuar(S/N)? ")
+   }
+   return (data.frame(nombres, telefonos))
+ }
> miagenda2<-add.agenda(miagenda)
Dime el nombre: Claudia
Dime el teléfono: 678456123
Continuar(S/N)? S
Dime el nombre: Miguel
Dime el teléfono: 678654456
Continuar(S/N)? N
> miagenda2
  nombres telefonos
1 Alejandro 678543234
2 Carlos 678987234
3 Maria 678345123
4 Roberto 678098987
5 Claudia 678456123
6 Miguel 678654456
```

Función construyepares, calcula los n pares consecutivos a un numero par m:



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

```
85 #Construir un vector que contenga los n numeros pares siguientes a m (par), m = 30
86
87 construyepares <- function(n, m) {
88   if (m%%2 == 1) { v <- "error" }
99:1   Añadir nuevos contactos y telefonos a "miagenda" }
```

The console shows the execution of the function:

```
> #Construir un vector que contenga los n numeros pares siguientes a m (par), m = 30
> construyepares <- function(n, m) {
+   if (m%%2 == 1) { v <- "error" }
+   else {
+     v <- m+2
+     m <- m+2
+     while (n>1)
+     {
+       m <- m+2
+       v <- c(v,m)
+       n <- n-1
+     }
+     return(v) }
>
> construyepares(7,30)
[1] 32 34 36 38 40 42 44
> |
```

The environment pane on the right shows the Global Environment with the following objects:

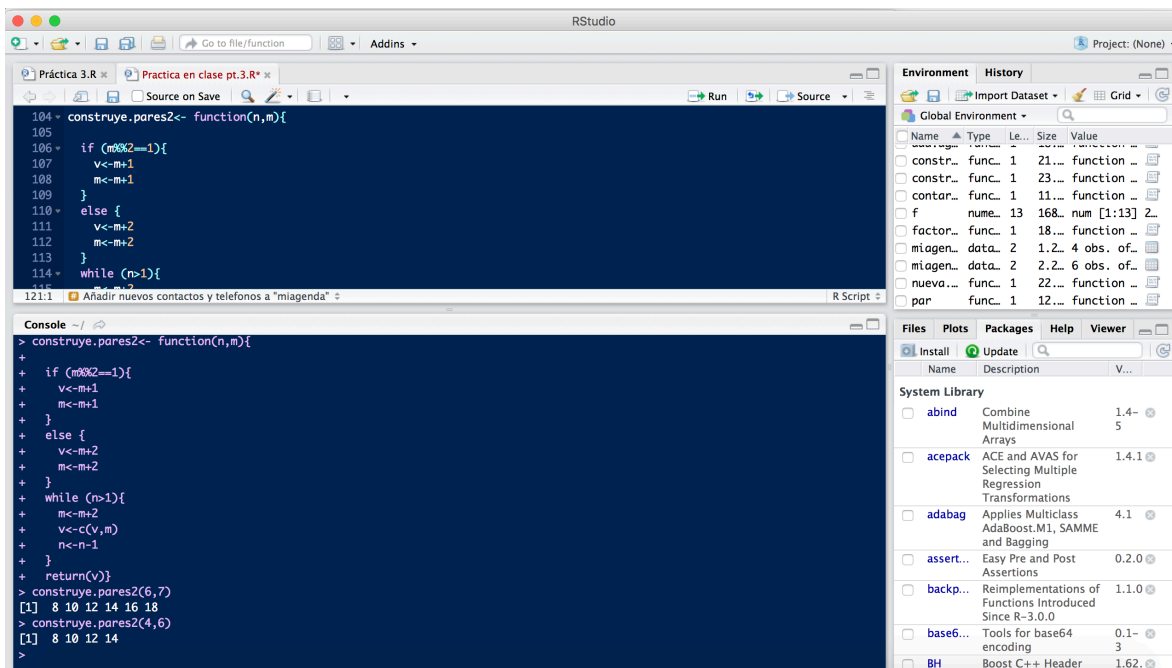
Name	Type	Le...	Size	Value
add.ag...	func...	1	18...	function ...
constr...	func...	1	21...	function ...
contar...	func...	1	11...	function ...
f	nume...	13	168...	num [1:13] 2...
factor...	func...	1	18...	function ...
miagen...	data...	2	1.2...	4 obs. of...
miagen...	data...	2	2.2...	6 obs. of...
nueva...	func...	1	22...	function ...
par	func...	1	12...	function ...

The Packages pane shows the System Library with the following packages:

Name	Description	V...
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
adabag	Applies Multiclass AdaBoost.M1, SAMME and Bagging	4.1
assert...	Easy Pre and Post Assertions	0.2.0
backp...	Reimplementations of Functions Introduced Since R-3.0.0	1.1.0
base6...	Tools for base64 encoding	0.1-3
BH	Boost C++ Header	1.62

Ejercicio 2.

Realizar las modificaciones pertinentes en el último ejemplo para que también funcione si el número m es impar, almacenar todo en un nuevo script y ejecutarlo para ver su funcionamiento. Para ello usar 'Interpretar código fuente' del menú de consola o ejecutar el comando: `source("ruta/fichero.R")`:



The screenshot shows the RStudio interface. The source editor on the left contains the following R code:

```
104 construye.pares2<- function(n,m){
105
106   if (m%2==1){
107     v<-m+1
108     m<-m+1
109   }
110   else {
111     v<-m+2
112     m<-m+2
113   }
114   while (n>1){
115     m<-m+2
116     v<-c(v,m)
117     n<-n-1
118   }
119   return(v)}
120
121 Añadir nuevos contactos y telefonos a "miagenda"
```

The console on the bottom left shows the execution of the function:

```
> construye.pares2<- function(n,m){
+   if (m%2==1){
+     v<-m+1
+     m<-m+1
+   }
+   else {
+     v<-m+2
+     m<-m+2
+   }
+   while (n>1){
+     m<-m+2
+     v<-c(v,m)
+     n<-n-1
+   }
+   return(v)}
> construye.pares2(6,7)
[1] 8 10 12 14 16 18
> construye.pares2(4,6)
[1] 8 10 12 14
>
```

The environment pane on the right shows the global environment with the following objects:

Name	Type	Le...	Size	Value
constr...	func...	1	21...	function ...
constr...	func...	1	23...	function ...
contar...	func...	1	11...	function ...
f	nume...	13	168...	num [1:13] 2...
factor...	func...	1	18...	function ...
mlagen...	data...	2	1.2...	4 obs. of...
mlagen...	data...	2	2.2...	6 obs. of...
nueva...	func...	1	22...	function ...
par	func...	1	12...	function ...

The packages pane on the right shows the system library with the following packages:

Name	Description	V...
abind	Combine Multidimensional Arrays	1.4-5
acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
adabag	Applies Multiclass AdaBoost.M1, SAMME and Bagging	4.1
assert...	Easy Pre and Post Assertions	0.2.0
backp...	Reimplementations of Functions Introduced Since R-3.0.0	1.1.0
base6...	Tools for base64 encoding	0.1-3
BH	Boost C++ Header	1.62.