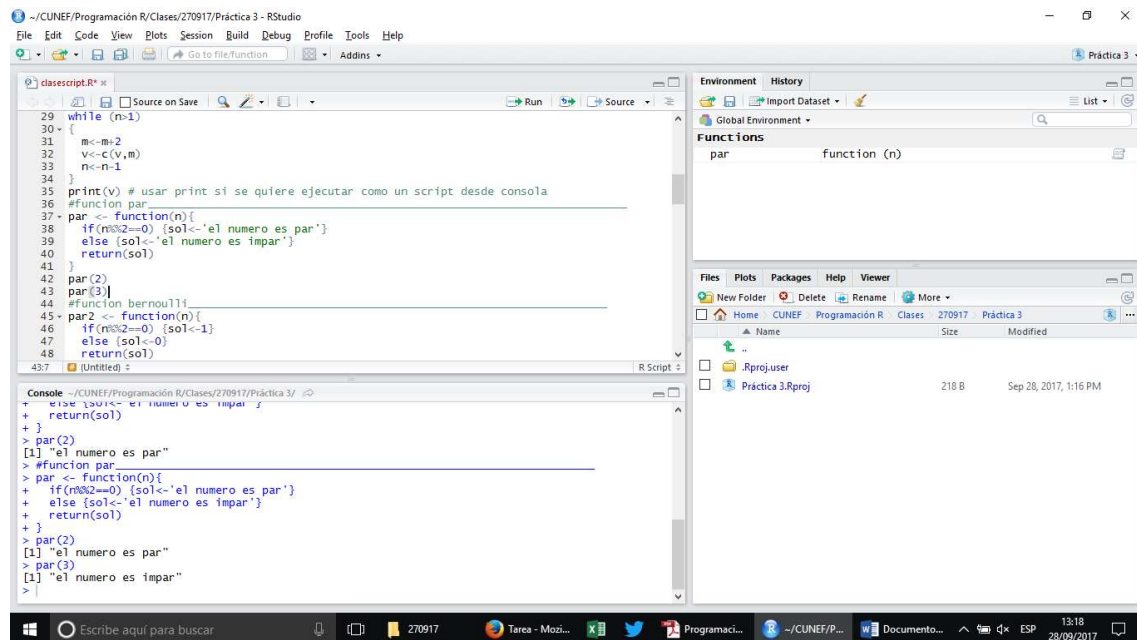


Programación con R. Práctica 3

Ejercicio 1. Programación de Scripts y Funciones.

Saber si un número es par

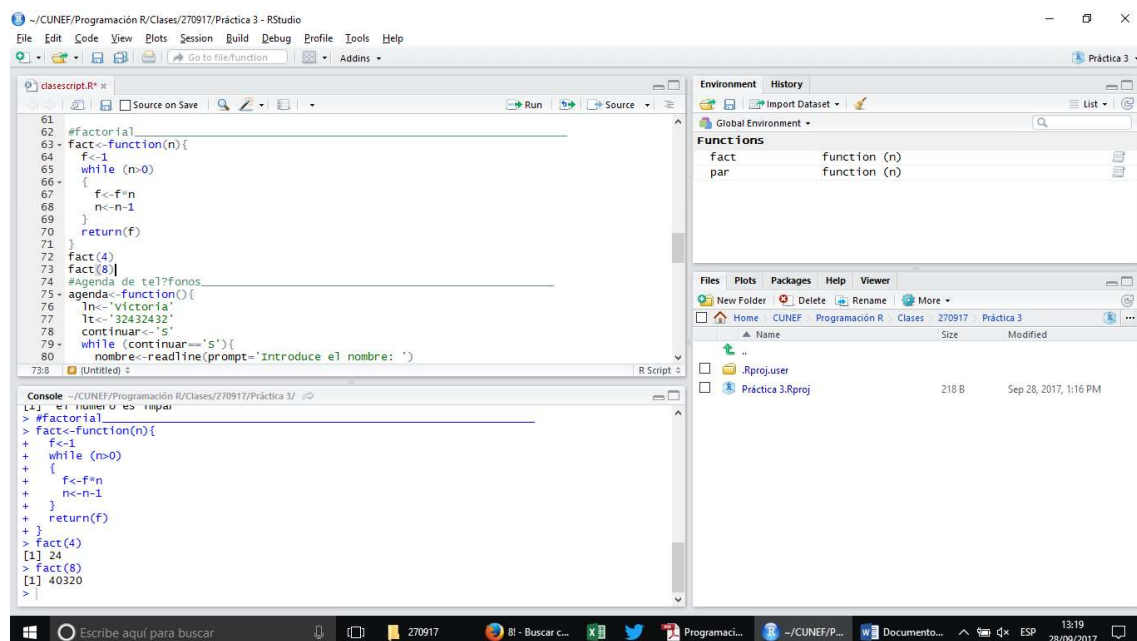


The screenshot shows the RStudio interface with a script file named 'clasescript.R'. The script defines a function 'par' that takes a number 'n' as input and returns a string indicating if it is even or odd. The console shows the execution of the function for inputs 2 and 3.

```
29 while (n>1)
30 {
31   m<-m+2
32   v<-c(v,m)
33   n<-n-1
34 }
35 print(v) # usar print si se quiere ejecutar como un script desde consola
36 #function par
37 par <- function(n){
38   if(n%%2==0) {sol<-'el numero es par'}
39   else {sol<-'el numero es impar'}
40   return(sol)
41 }
42 par(2)
43 par(3)
44 #Function bernoulli
45 par2 <- function(n){
46   if(n%%2==0) {sol<-'1'}
47   else {sol<-'0'}
48   return(sol)
49 }
```

```
> par(2)
[1] "el numero es par"
> #function par
> par <- function(n){
+   if(n%%2==0) {sol<-'el numero es par'}
+   else {sol<-'el numero es impar'}
+   return(sol)
+ }
> par(2)
[1] "el numero es par"
> par(3)
[1] "el numero es impar"
>
```

Calcular el factorial de un número



The screenshot shows the RStudio interface with a script file named 'clasescript.R'. The script defines a function 'factorial' that takes a number 'n' as input and returns the factorial of 'n'. The console shows the execution of the function for inputs 4 and 8.

```
61 #factorial
62 fact<-function(n){
63   f<-1
64   while (n>0)
65   {
66     f<-f*n
67     n<-n-1
68   }
69   return(f)
70 }
71 fact(4)
72 fact(8)
73 #Agenda de teléfonos
74 agenda<-function(){
75   ln<-'Victoria'
76   lt<-'32432432'
77   continuar<-'s'
78   while (continuar=='s'){
79     nombre<-readline(prompt='Introduce el nombre: ')
80   }
81 }
```

```
> #factorial
> fact<-function(n){
+   while (n>0)
+   {
+     f<-f*n
+     n<-n-1
+   }
+   return(f)
+ }
> fact(4)
[1] 24
> fact(8)
[1] 40320
>
```

Crear una agenda con nombres y teléfonos de tres amigos

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R script for the `agenda` function. The function prompts the user to enter a name and a phone number, stores them in a data frame, and returns the data frame. It includes a `while` loop for continuous input.
- Console:** Shows the execution of the `agenda` function. It prompts for three entries: Abraham (4534543), Luis (4534534), and Jose (4665756756). The output is a data frame with 4 rows (including the header) and 2 columns (`ln` and `lt`).
- Environment:** Lists the functions `agenda`, `fact`, and `par`.
- Files:** Shows the project structure with files like `.Rproj.user` and `Práctica 3.Rproj`.

```
# Agenda de teléfonos
agenda<-function(){
  ln<-"Victoria"
  lt<-"32432432"
  continuar<-"s"
  while (continuar=="s"){
    nombre<-readline(prompt="Introduce el nombre: ")
    telefono<-readline(prompt="Introduce el telefono: ")
    ln<-c(ln,nombre)
    lt<-c(lt,telefono)
    continuar<- readline(prompt="continuar?S/N:")
  }
  mdf<-data.frame(ln,lt)
  mdf$ln<- as.character(mdf$ln)
  mdf$lt<- as.character(mdf$lt)
  return (mdf)
}
agenda()
```

```
> agenda()
Introduce el nombre: Abraham
Introduce el telefono: 4534543
Continuar?S/N:s
Introduce el nombre: Luis
Introduce el telefono: 4534534
Continuar?S/N:s
Introduce el nombre: Jose
Introduce el telefono: 4665756756
Continuar?S/N:N
  ln      lt
1 Victoria 32432432
2 Abraham  4534543
3 Luis     4534534
4 Jose     4665756756
```

Añadir dos amigos más a la agenda

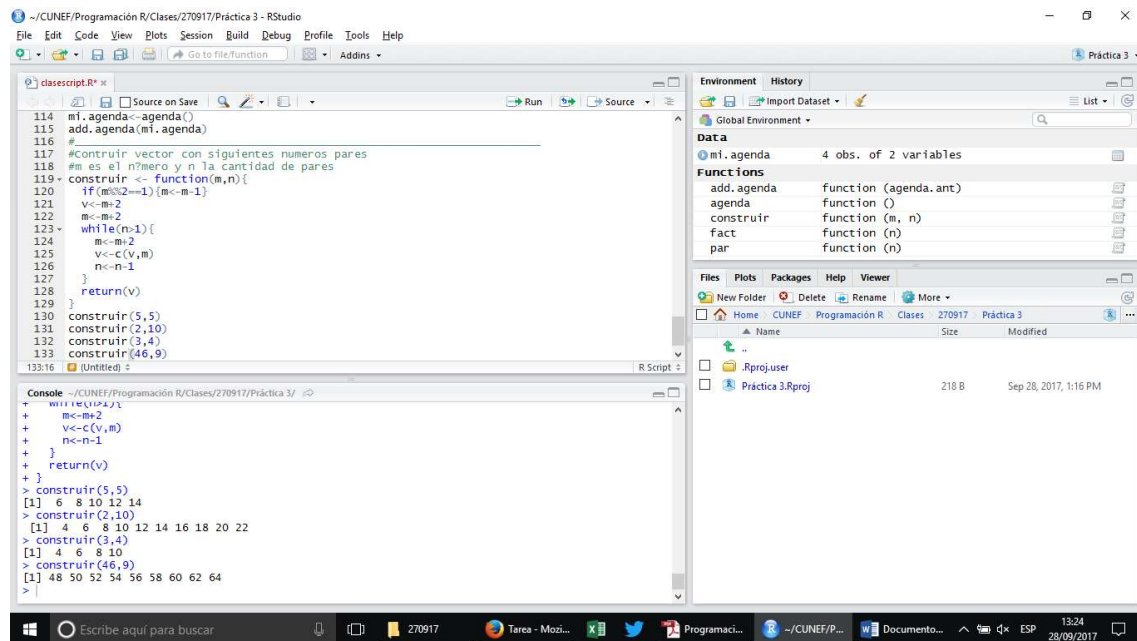
The screenshot shows the RStudio interface after adding more friends to the agenda. The `add.agenda` function is used to append new entries to the existing data frame.

- Source Editor:** Shows the `add.agenda` function, which takes an existing agenda (data frame) and a new entry (name and phone number) as input, and returns the updated data frame.
- Console:** Shows the execution of `add.agenda(mi.agenda)`. It prompts for two more entries: Alejandra (67464534) and Ivonne (676567563). The output shows the updated data frame with 6 rows.
- Environment:** Lists the functions `add.agenda`, `agenda`, `fact`, and `par`.
- Files:** Shows the project structure with files like `.Rproj.user` and `Práctica 3.Rproj`.

```
add.agenda<-function(agenda.ant){
  # esta función añade nombres y teléfonos a una agenda que está almacenada
  # en un data frame con dos campos que son nombres y teléfonos.
  ln<-agenda.ant$ln
  lt<-agenda.ant$lt
  continuar<-"s"
  while (continuar=="s"){
    nombre<-readline(prompt="Introduce el nombre: ")
    telefono<-readline(prompt="Introduce el telefono: ")
    ln<-c(ln,nombre)
    lt<-c(lt,telefono)
    continuar<- readline(prompt="continuar?S/N:")
  }
  return (data.frame(ln,lt))
}
mi.agenda<-agenda()
add.agenda(mi.agenda)
```

```
> add.agenda(mi.agenda)
Introduce el nombre: Alejandra
Introduce el telefono: 67464534
Continuar?S/N:s
Introduce el nombre: Ivonne
Introduce el telefono: 676567563
Continuar?S/N:N
  ln      lt
1 Victoria 32432432
2 Abraham  345345345
3 Luis     5654654
4 Jose     4534665756
5 Alejandra 67464534
6 Ivonne    676567563
```

Función dónde se introduce un número m y regresa los siguientes n números pares, independientemente de si m es par o impar



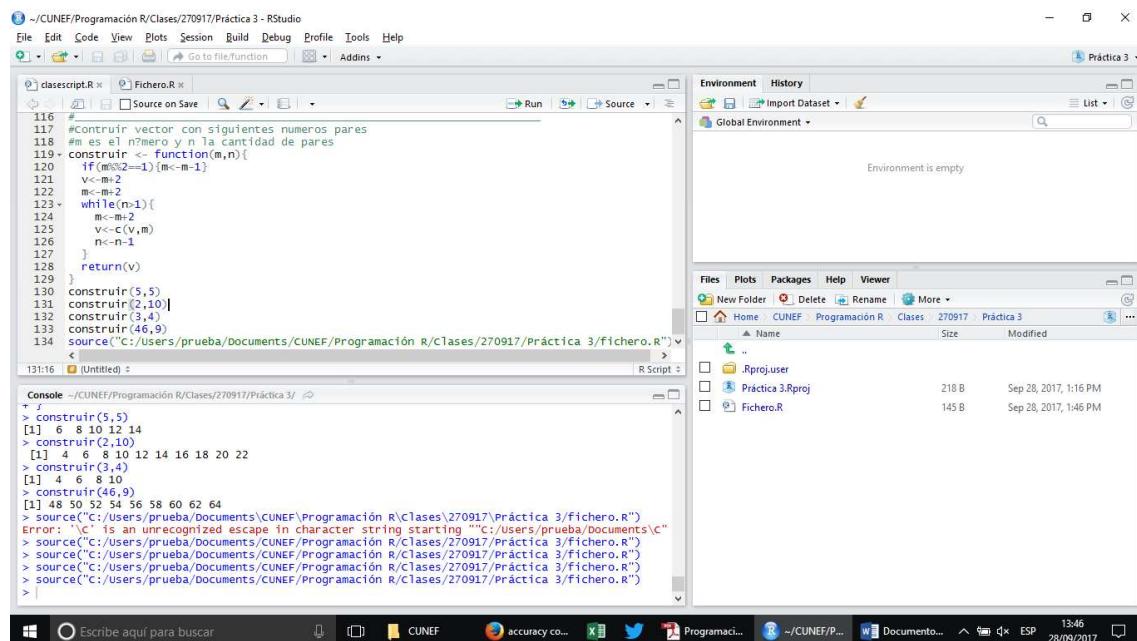
The screenshot shows the RStudio interface with a script file named 'clasescript.R'. The script defines a function 'construir' that takes two arguments, 'm' and 'n', and returns a vector of 'n' even numbers starting from 'm'. The function uses a while loop to generate the sequence. The console shows the execution of the function for various inputs: 'construir(5,5)' returns [1] 6 8 10 12 14; 'construir(2,10)' returns [1] 4 6 8 10 12 14 16 18 20 22; 'construir(3,4)' returns [1] 4 6 8 10; and 'construir(46,9)' returns [1] 48 50 52 54 56 58 60 62 64. The Environment pane on the right shows the global environment with variables 'mi.agenda' (4 obs. of 2 variables) and functions 'add.agenda', 'agenda', 'construir', 'fact', and 'par'.

```
114 mi.agenda<-agenda()
115 add.agenda(mi.agenda)
116 #
117 #Construir vector con siguientes numeros pares
118 #m es el n?mero y n la cantidad de pares
119 construir <- function(m,n){
120   if(m%%2==1){m<-m-1}
121   v<-m+2
122   m<-m+2
123   while(n>1){
124     m<-m+2
125     v<-c(v,m)
126     n<-n-1
127   }
128   return(v)
129 }
130 construir(5,5)
131 construir(2,10)
132 construir(3,4)
133 construir(46,9)
133:16 [Untitled] R Script
```

```
> construir(5,5)
[1] 6 8 10 12 14
> construir(2,10)
[1] 4 6 8 10 12 14 16 18 20 22
> construir(3,4)
[1] 4 6 8 10
> construir(46,9)
[1] 48 50 52 54 56 58 60 62 64
>
```

Ejercicio 2.

Observamos que no tenemos ningún objeto en nuestro workspace:



The screenshot shows the RStudio interface with a script file named 'Fichero.R'. The script defines the same 'construir' function as in the previous screenshot. The console shows the execution of the function for various inputs, followed by an error message: 'Error: '\c' is an unrecognized escape in character string starting ""C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R"'. The Environment pane on the right shows the global environment with variables '.Rproj.user' and 'Práctica 3.Rproj'.

```
116 #
117 #Construir vector con siguientes numeros pares
118 #m es el n?mero y n la cantidad de pares
119 construir <- function(m,n){
120   if(m%%2==1){m<-m-1}
121   v<-m+2
122   m<-m+2
123   while(n>1){
124     m<-m+2
125     v<-c(v,m)
126     n<-n-1
127   }
128   return(v)
129 }
130 construir(5,5)
131 construir(2,10)
132 construir(3,4)
133 construir(46,9)
134 source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
131:16 [Untitled] R Script
```

```
> construir(5,5)
[1] 6 8 10 12 14
> construir(2,10)
[1] 4 6 8 10 12 14 16 18 20 22
> construir(3,4)
[1] 4 6 8 10
> construir(46,9)
[1] 48 50 52 54 56 58 60 62 64
> source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
Error: '\c' is an unrecognized escape in character string starting ""C:/Users/prueba/documents/c"
> source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
> source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
> source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
> source("C:/Users/prueba/documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")
>
```

Al ejecutar la sentencia `source("C:/Users/prueba/Documents/CUNEF/Programación R/Clases/270917/Práctica 3/fichero.R")` nos carga todas las funciones que están en un script diferente

