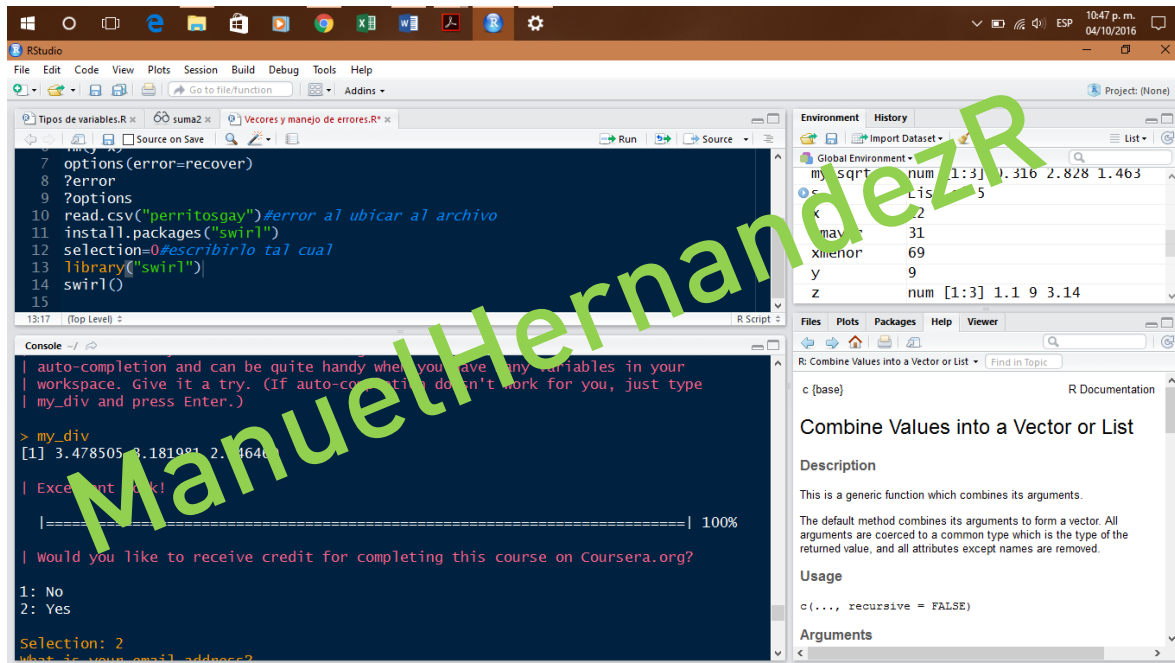


Mis cursos terminados

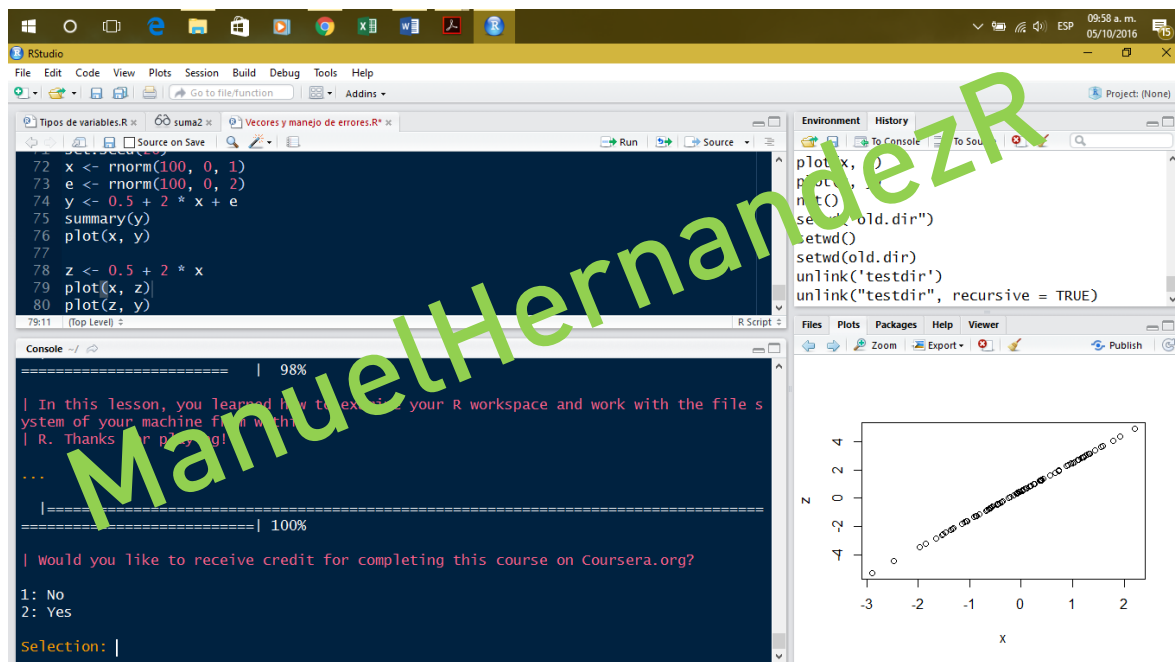
Por: Hernández Romero, José Manuel

201559002

1.- Basic Building Blocks



2.- Workspaces and Files



3.- Sequences of Numbers

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

```
88 plot(x, y, main = "Modelo Lineal")
89
90 set.seed(1)
91 x <- rnorm(100)
```

The console shows a progress bar at 96% and a survey question: "Would you like to receive credit for completing this course on Coursera.org?". The user has selected "No".

The Environment pane on the right shows the following objects:

```
length(my_seq)
length(my_seq)
1: length(my_seq)
seq(alongwith=my_seq)
seq_along(my_seq)
rep(0, times=40)
rep(c(0, 1, 2), times = 10)
rep(c(0, 1, 2), each = 10)
```

The Help pane on the right shows the documentation for the `seq` function, including examples of its usage.

4.- Numbers

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

```
88 plot(x, y, main = "Modelo Lineal")
89
90 set.seed(1)
91 x <- rnorm(100)
```

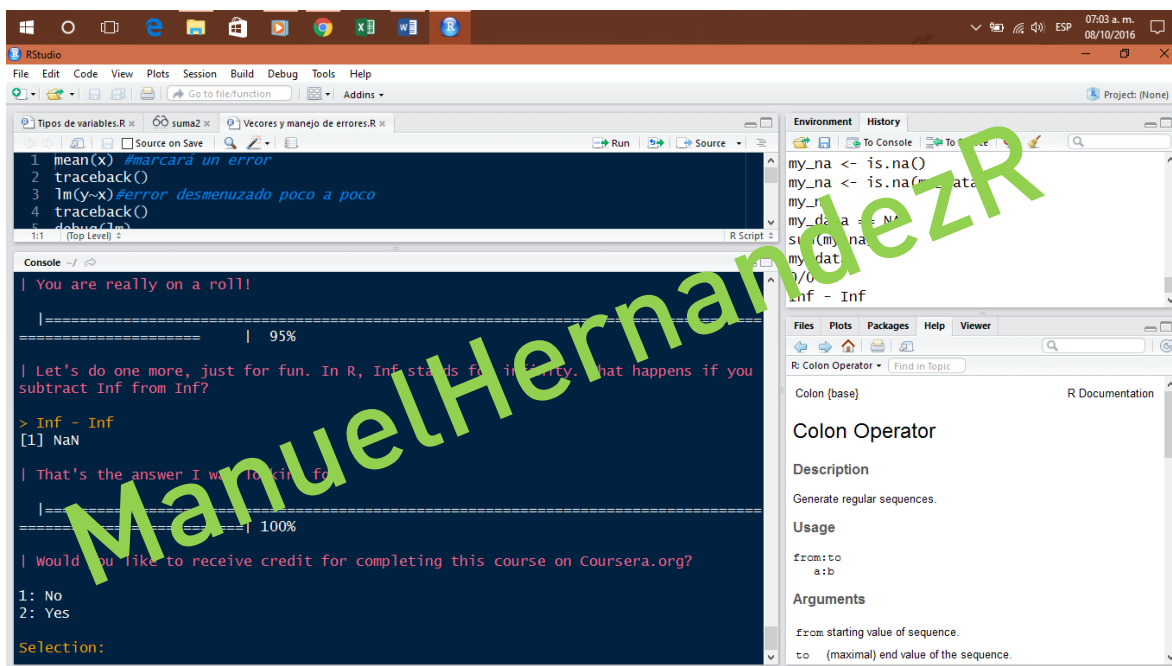
The console shows a progress bar at 97% and a survey question: "Would you like to receive credit for completing this course on Coursera.org?". The user has selected "No".

The Environment pane on the right shows the following objects:

```
c(my_char, "Manuel Hernández")
my_name <- c(my_char, "Manuel Hernández")
my_name
paste(my_name, "lap" = " ")
paste("Hel", "ld", sep = " ")
paste(c("X", "Y", "Z"), sep = " ")
paste(1:3, c("X", "Y", "Z"), sep = " ")
paste(LETTERS, 1:4, sep = "-")
```

The Help pane on the right shows the documentation for the `seq` function, including examples of its usage.

5.- Missing Values:



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

```
1 mean(x) #marcará un error
2 traceback()
3 lm(y~x) #error desmenuzado poco a poco
4 traceback()
```

The console shows the following output:

```
| You are really on a roll!

===== | 95%

| Let's do one more, just for fun. In R, Inf stands for infinity. What happens if you
subtract Inf from Inf?

> Inf - Inf
[1] NaN

| That's the answer I was looking for

===== | 100%

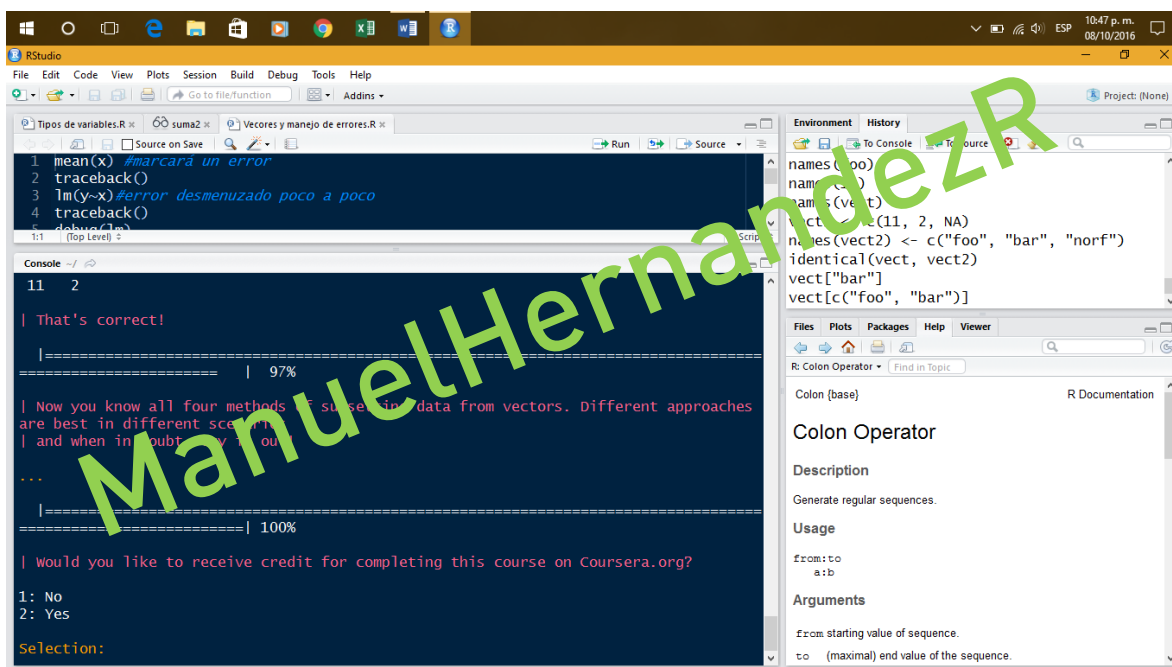
| Would you like to receive credit for completing this course on Coursera.org?
1: No
2: Yes
Selection:
```

The Environment pane on the right shows the following objects:

```
my_na <- is.na()
my_na <- is.na(my_data)
my_data <- NA
my_data[1:10] <- NA
my_data[11] <- Inf
my_data[12] <- Inf
```

The Help pane on the right shows the documentation for the Colon Operator.

6.- Subsetting colors:



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

```
1 mean(x) #marcará un error
2 traceback()
3 lm(y~x) #error desmenuzado poco a poco
4 traceback()
```

The console shows the following output:

```
11 2

| That's correct!

===== | 97%

| Now you know all four methods of subsetting data from vectors. Different approaches
are best in different scenarios.
| and when in doubt, try them all.

...

===== | 100%

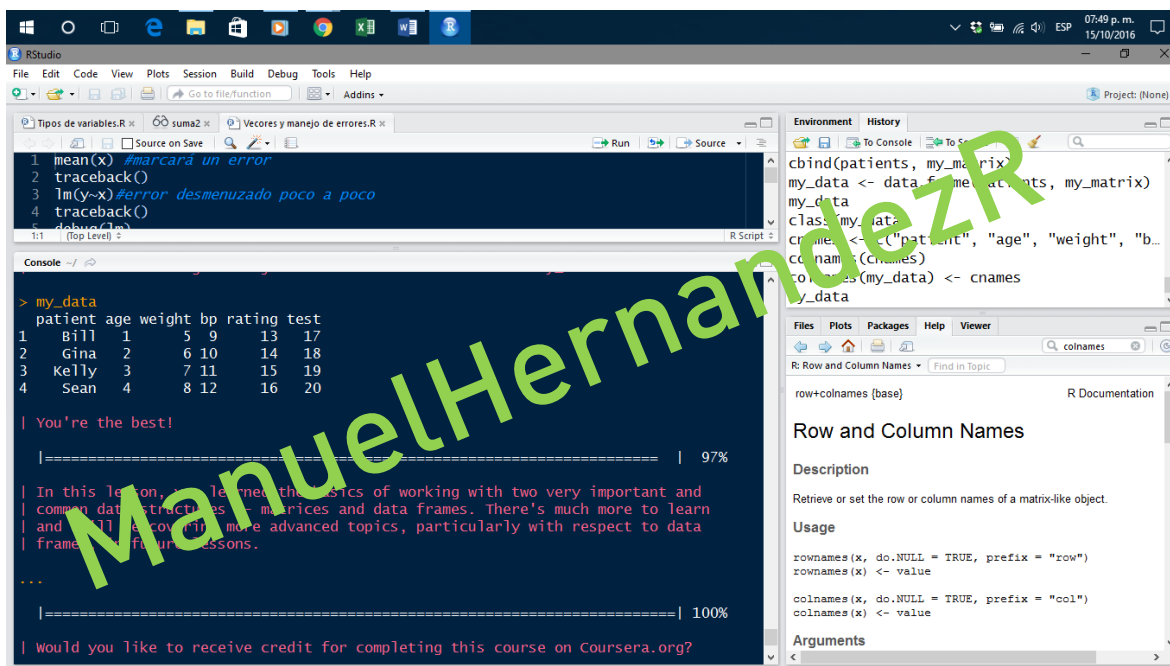
| Would you like to receive credit for completing this course on Coursera.org?
1: No
2: Yes
Selection:
```

The Environment pane on the right shows the following objects:

```
names(vect)
names(vect2)
names(vect3)
vect <- c(11, 2, NA)
names(vect2) <- c("foo", "bar", "norf")
identical(vect, vect2)
vect["bar"]
vect[c("foo", "bar")]
```

The Help pane on the right shows the documentation for the Colon Operator.

7.- Matrices and data frames:



```
1 mean(x) #marcará un error
2 traceback()
3 lm(y~x) #error desmenuzado poco a poco
4 traceback()
5 #Answer 4/5
```

```
> my_data
  patient age weight bp rating test
1 Bill 1 5 9 13 17
2 Gina 2 6 10 14 18
3 Kelly 3 7 11 15 19
4 Sean 4 8 12 16 20
```

| You're the best!

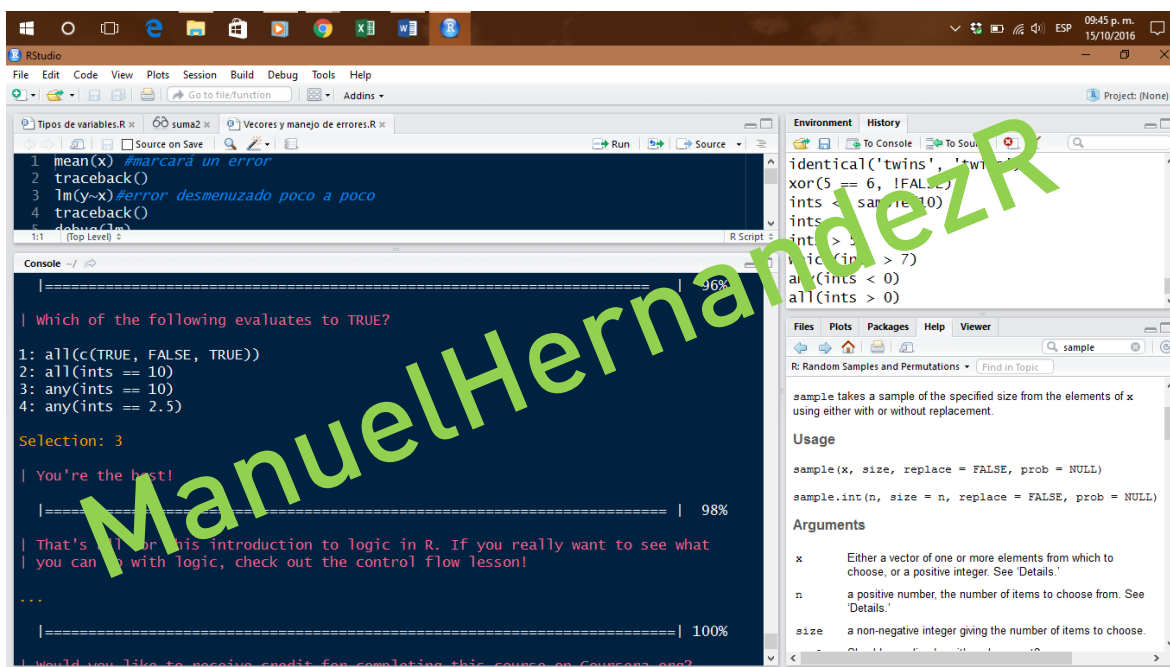
| In this lesson, you learned the basics of working with two very important and common data structures - matrices and data frames. There's much more to learn and we'll cover more advanced topics, particularly with respect to data frames, in future lessons.

| would you like to receive credit for completing this course on Coursera.org?

Environment

```
cbind(patients, my_matrix)
my_data <- data.frame(patients, my_matrix)
my_data
class(my_data)
cnames <- c("patient", "age", "weight", "b...
colnames(my_data) <- cnames
my_data
```

8.- Logic:



```
1 mean(x) #marcará un error
2 traceback()
3 lm(y~x) #error desmenuzado poco a poco
4 traceback()
5 #Answer 4/5
```

```
1: all(c(TRUE, FALSE, TRUE))
2: all(ints == 10)
3: any(ints == 10)
4: any(ints == 2.5)
```

Selection: 3

| You're the best!

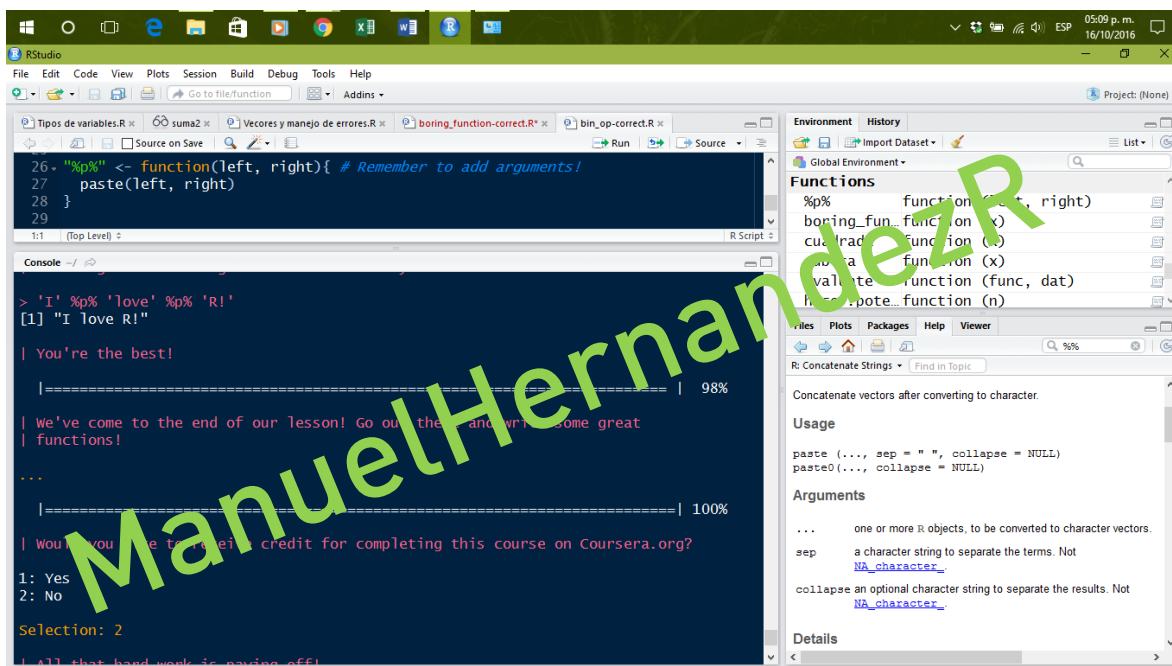
| That's all for this introduction to logic in R. If you really want to see what you can do with logic, check out the control flow lesson!

| would you like to receive credit for completing this course on Coursera.org?

Environment

```
identical('twins', 'twins')
xor(5 == 6, FALSE)
ints <- sample(10)
ints
ints > 5
which(ints > 7)
any(ints < 0)
all(ints > 0)
```

9.- Functions:



The screenshot shows the RStudio interface. The script editor contains a function definition for `paste` with a comment: `# Remember to add arguments!`. The console shows the function being called with arguments: `> 'I' %p% 'love' %p% 'R!'`, resulting in `[1] "I love R!"`. The progress bar indicates 98% completion. The Environment pane on the right shows the function definition and its usage.

```
26. "%p%" <- function(left, right){ # Remember to add arguments!
27.   paste(left, right)
28. }
29.
```

```
> 'I' %p% 'love' %p% 'R!'
[1] "I love R!"
```

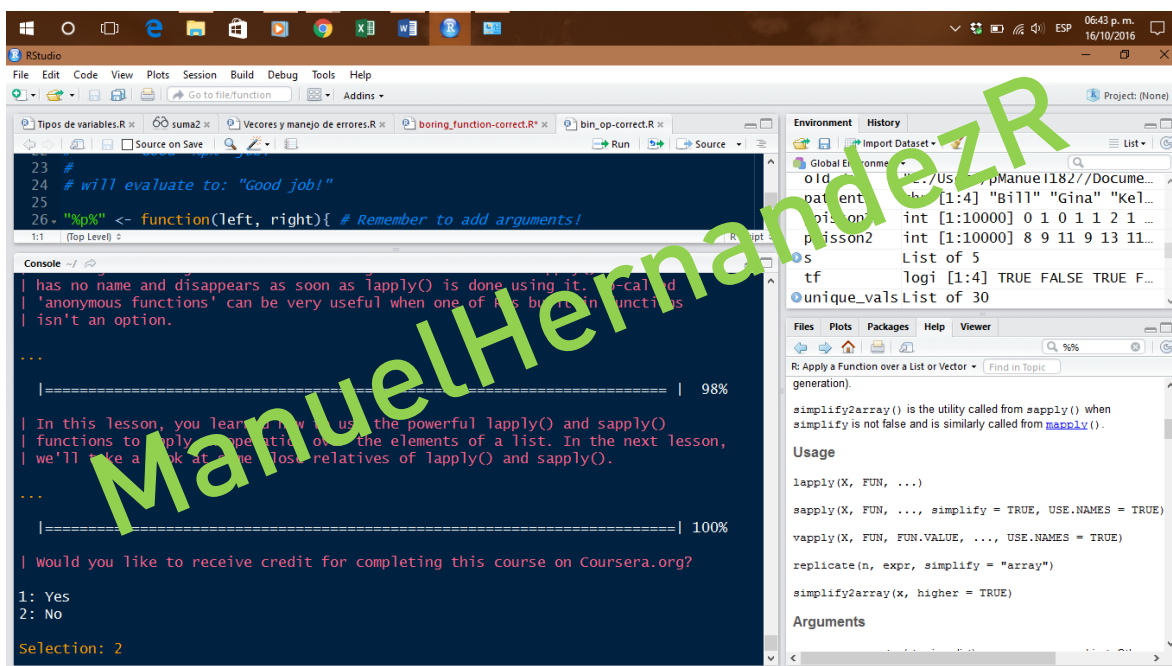
Usage

```
paste(..., sep = " ", collapse = NULL)
paste0(..., collapse = NULL)
```

Arguments

- ... one or more R objects, to be converted to character vectors.
- sep a character string to separate the terms. Not `NA_character_`.
- collapse an optional character string to separate the results. Not `NA_character_`.

10.- Lapply and sapply:



The screenshot shows the RStudio interface. The script editor contains a function definition for `lapply` with a comment: `# Remember to add arguments!`. The console shows the function being called with arguments: `> 'I' %p% 'love' %p% 'R!'`, resulting in `[1] "I love R!"`. The progress bar indicates 98% completion. The Environment pane on the right shows the function definition and its usage.

```
23. #
24. # will evaluate to: "Good job!"
25.
26. "%p%" <- function(left, right){ # Remember to add arguments!
```

```
> 'I' %p% 'love' %p% 'R!'
[1] "I love R!"
```

Usage

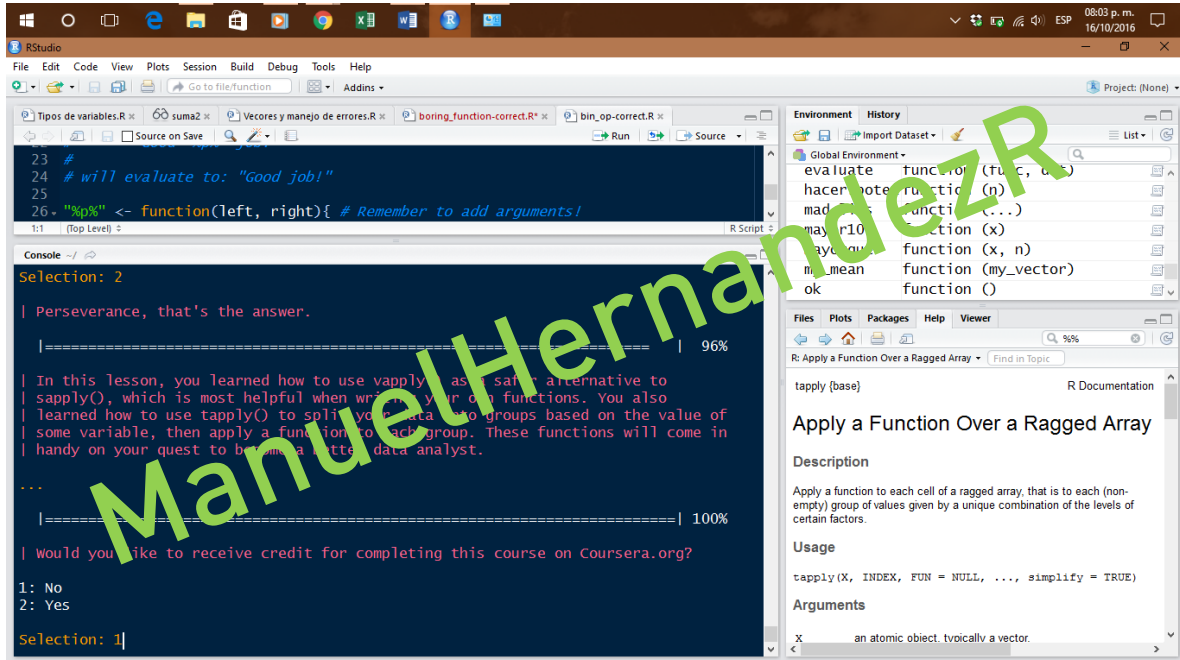
```
lapply(X, FUN, ...)
```

```
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

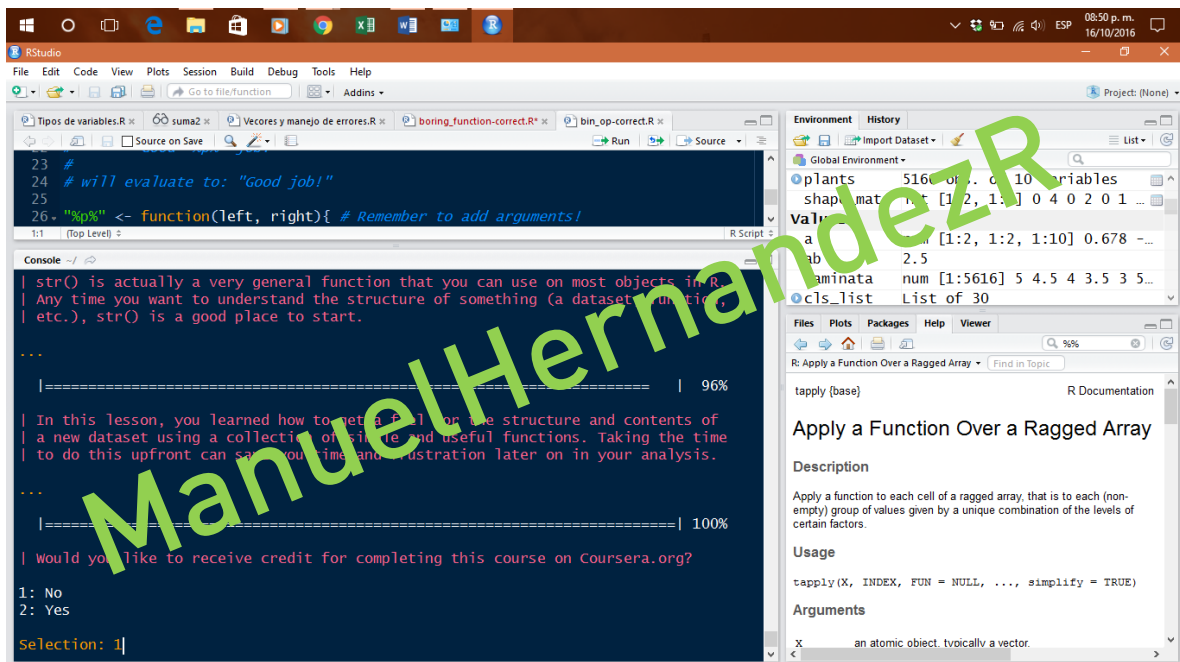
```
replicate(n, expr, simplify = "array")
```

```
simplify2array(x, higher = TRUE)
```

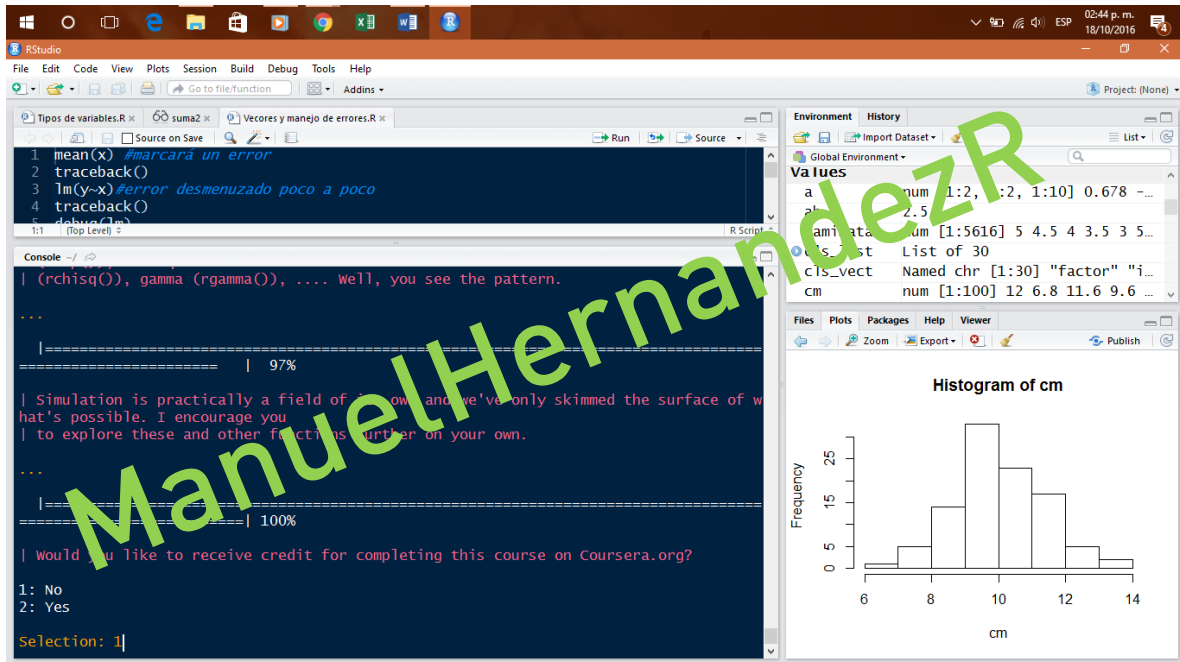
11.- Vapply and tapply()



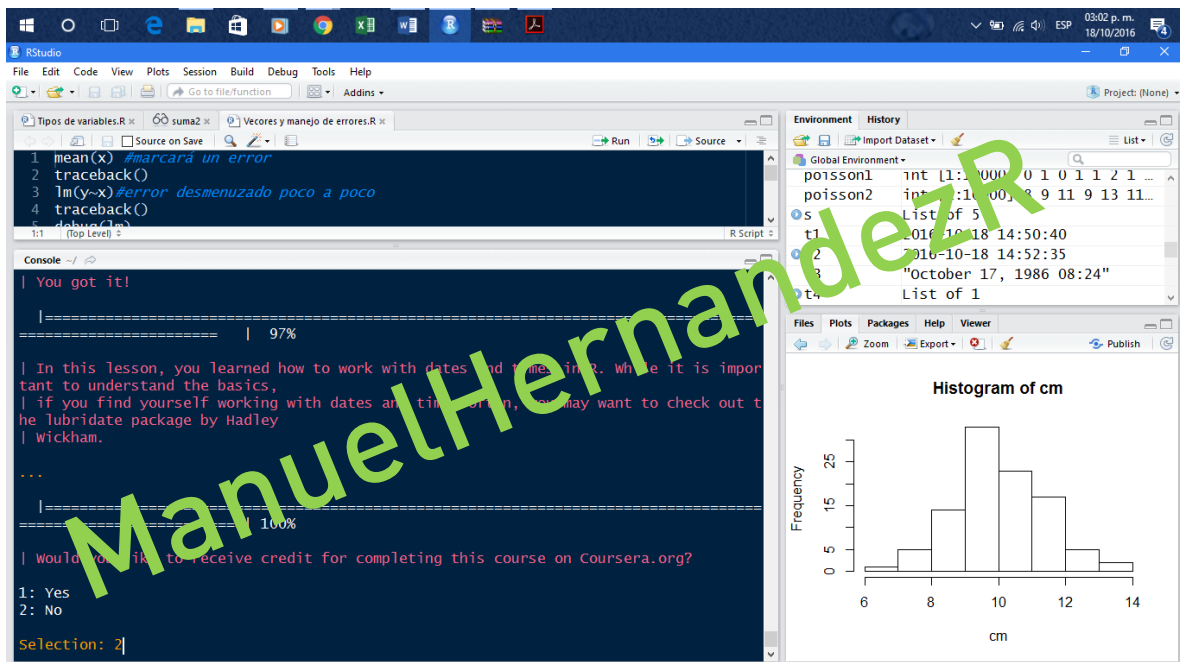
12.- Looking at data



13.- Simulation



14.- Dates and times:



15.- Base graphics:

