

The RStudio interface displays a script for data manipulation. The script includes the following code:

```

104 # Split on data frame
105 library(datasets)
106 head(airquality)
107 print(airquality)
108 split(airquality, airquality$month)
109
110 s <- split(airquality, airquality$month)
111 lapply(s, function(x) colMeans(x[,1:3], na.rm = T))
112 # Como nos da algo que podemos recorrer con, ocupamos lapply, para simplificarlo
113 lapply(s, function(x) colMeans(x[,1:3], na.rm = T))
114
115 # cuando hay errores
116 traceback() # esto muestra como fue corriendo y cual fue el ultimo paso ejecutado
117 # para ver como bien
118 debug(hacer.potencia) # para ir paso a paso hasta encontrar el error, con "n" cambia
119
120 # cuando hay error
121 # con select() marca donde quiere verificar
122
123
124

```

The console shows the output of the script:

```

> my_d1v
[1] 1.478105 1.181941 2.140460

# excellent work!

===== 100%

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

1) No
2) Yes

```

The help window for the `c` function is open, showing the following information:

Combine Values into a Vector or List

Description

This is a generic function which combines its arguments.

The default method combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the named value, and all attributes except names are removed.

Usage

```
c(..., recursive = FALSE)
```

Arguments

... objects to be concatenated

The RStudio interface displays a script for file system operations. The script includes the following code:

```

1 # to: test
2 # to: test
3 # to: test
4 # to: test
5 # to: test
6 # to: test
7 # to: test
8
9 # library(stats)
10 search()
11 searchpaths()
12
13 hacer.potencia <- function(n){
14   potencia <- function(x){
15     x = x
16   }
17   potencia
18 }
19 hacer.potencia()
20
21 cubica <- hacer.potencia(1)
22 cuadrada <- hacer.potencia(2)
23 cubica
24

```

The console shows the output of the script:

```

[ In this lesson, you learned how to examine your R workspace and work with the file system of your machine from within R. Thanks for playing!

===== 100%

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

1) YES
2) No
Selection:

```

The help window for the `dir.exists` function is open, showing the following information:

dir.exists (paths)

dir.exists(paths, showWarnings = TRUE, recursive = FALSE, x = Sys.getenv("PATH"), mode = "rwx", use_unlink = TRUE)

Arguments

paths a character vector containing a single path name. Wildcard expansion (see [path.expand](#)) is done.

paths character vectors containing file or directory paths. Wildcard expansion (see [path.expand](#)) is done.

showWarnings logical: should the warnings on false be shown?

recursive logical: Should elements of the path other than the last be created? If true, like the Unix command mkdir -p.

mode the mode to be used as Unix allows; it will be coerced by [as.character](#). For Sys.getenv it is recycled along paths.

use_unlink logical: should the mode be restricted by the unlink.

The RStudio interface displays a script with the following code:

```

1 #26. seg
2 #m
3 #m <- function(x){x*x}
4 #m
5 rm(m)
6 #m <- function(x){x*x}
7 search()
8
9 library(stats)
10 search()
11 searchpaths()
12
13 #hacer.potencia <- function(n){
14 #  potencia <- function(x){
15 #    x = n
16 #  }
17 #  potencia
18 #}
19 #hacer.potencia(3)
20
21 cubica <- hacer.potencia(3)
22 cuadrada <- hacer.potencia(2)
23 cubica/2
24 #m
25 #m

```

The console shows the output of the 'inf-inf' command:

```

> inf-inf
[1] NaN

# nice work!

===== 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:

Object	Class	Attributes
my_data	num	[1:100] NA NA 0.0464 -1.8149 -0.0...
my_na	logi	[1:100] TRUE TRUE FALSE FALSE...
my_name	chr	[1:4] "my" "name" "is" "Emmanuel..."
my_seq	num	[1:20] 5 5.1 5.84 5.52 5.69 ...
my_vect	num	[1:4] 0.1 15 -10 8
my_vect1	logi	[1:4] FALSE TRUE FALSE TRUE
old_dir	chr	"C:/Users/Proprietario/Documents"

The Colon Operator documentation is visible on the right side of the interface.

The RStudio interface displays a script with the following code:

```

1 #26. seg
2 #m
3 #m <- function(x){x*x}
4 #m
5 rm(m)
6 #m <- function(x){x*x}
7 search()
8
9 library(stats)
10 search()
11 searchpaths()
12
13 #hacer.potencia <- function(n){
14 #  potencia <- function(x){
15 #    x = n
16 #  }
17 #  potencia
18 #}
19 #hacer.potencia(3)
20
21 cubica <- hacer.potencia(3)
22 cuadrada <- hacer.potencia(2)
23 cubica/2
24 #m
25 #m

```

The console shows the output of the 'inf-inf' command:

```

> inf-inf
[1] NaN

# nice work!

===== 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:

Object	Class	Attributes
my_data	num	[1:100] NA NA 0.0464 -1.8149 -0.0...
my_na	logi	[1:100] TRUE TRUE FALSE FALSE...
my_name	chr	[1:4] "my" "name" "is" "Emmanuel..."
my_seq	num	[1:20] 5 5.1 5.84 5.52 5.69 ...
my_vect	num	[1:4] 0.1 15 -10 8
my_vect1	logi	[1:4] FALSE TRUE FALSE TRUE
old_dir	chr	"C:/Users/Proprietario/Documents"
test_dir	chr	"C:/Users/Proprietario/Documents"
tf	logi	[1:4] TRUE FALSE TRUE FALSE
my_vect	num	[1:2] 11 2 NA
my_vect2	num	[1:2] 11 2 NA

The Colon Operator documentation is visible on the right side of the interface.

The RStudio interface displays a script with the following code:

```

239
240- h1art <- function(n){
241-   i <- 1:n
242-   i[outer(1:1,i,"*")]
243- }
244 x <- h1art(1000)
245 system.time: x <- h1art(1000)
246 system.time: print(x)
247 #don't() we dice cada .02 segundos en que función trabaja
248 #we star system y apraf porque se puede clicar...
249
250
251
252
253
254
255
256
257
258
259

```

The console output shows the progress of the function execution:

```

...
=====] 100%
/ would you like to receive credit for completing this course on coursera.org?
1) YES
2) NO
Selection:

```

The Environment pane on the right shows the following objects:

- Global Environment**
 - my_data**: 6 obs. of 8 variables
 - my_matrix**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
 - my_matrix2**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
 - my_vector**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...

The console also displays the following text:

```

matrix creates a matrix from the given set of values.
- It also attempts to turn its argument into a matrix.
- i.e. matrix tests if its argument is a (strict) matrix.
Usage
matrix(data = NA, nrow = 1, mcol = 1, byrow = FALSE,
        dimnames = NULL)
as.matrix(x, ...)
## 33 methods for class "data.frame"
as.matrix(x, rownames, colnames, force = NA, ...)
is.matrix(x)
Arguments
data
an optional data vector (including a list or environment vector). Non-atomic
classed R objects are coerced by as.vector and all attributes discarded.

```

Emmanuel Campos

The RStudio interface displays a script with the following code:

```

239
240- h1art <- function(n){
241-   i <- 1:n
242-   i[outer(1:1,i,"*")]
243- }
244 x <- h1art(1000)
245 system.time: x <- h1art(1000)
246 system.time: print(x)
247 #don't() we dice cada .02 segundos en que función trabaja
248 #we star system y apraf porque se puede clicar...
249
250
251
252
253
254
255
256
257
258
259

```

The console output shows the progress of the function execution:

```

...
=====] 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:

- Global Environment**
 - my_data**: 6 obs. of 8 variables
 - my_matrix**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
 - my_matrix2**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
 - my_vector**: int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...

The console also displays the following text:

```

matrix creates a matrix from the given set of values.
- It also attempts to turn its argument into a matrix.
- i.e. matrix tests if its argument is a (strict) matrix.
Usage
matrix(data = NA, nrow = 1, mcol = 1, byrow = FALSE,
        dimnames = NULL)
as.matrix(x, ...)
## 33 methods for class "data.frame"
as.matrix(x, rownames, colnames, force = NA, ...)
is.matrix(x)
Arguments
data
an optional data vector (including a list or environment vector). Non-atomic
classed R objects are coerced by as.vector and all attributes discarded.

```

Emmanuel Campos

Script Editor:

```

1 # You're about to write your first function! Just like you would assign a value
2 # to a variable with the assignment operator, you assign functions to the following
3 # way:
4 #
5 # function_name <- function(arg1, arg2){
6 #   # manipulate arguments in some way
7 #   # return a value
8 # }
9 #
10 # the "variable name" you assign will become the name of your function, arg1 and
11 # arg2 represent the arguments of your function. You can manipulate the arguments
12 # you specify within the function. After sourcing the function, you can use the
13 # function by typing:
14 #
15 # function_name(value1, value2)
16 #
17 # Now we will create a function called boring_function. This function takes
18 # the argument 'x' as input, and returns the value of 'x' without modifying it.
19 # Delete the pound sign in front of the x to make the function work! Be sure to
20 # save this script and type submit() in the console after you make your changes.
21 #
22 boring_function <- function(x){
23   x
24 }
  
```

Console:

```

1 on your quest to become a better data analyst.
...
=====| 100%
1) would you like to receive credit for completing this course on coursera.org?
2) YES
3) NO
Selection:
  
```

Environment:

Object	Class
ok	Function ()
visualinfo	Function ()

Help Window: apply

Apply a Function Over a Ragged Array

Description
Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Usage
apply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

Arguments
X an atomic object, typically a vector
INDEX list of one or more factors, each of same length as X. The elements are coerced to factors by as.factor

Script Editor:

```

1 function ok, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
2 {
3   FUN <- match.fun(FUN)
4   answer <- apply(X = X, FUN = FUN, ...)
5   if (USE.NAMES && is.character(X) && is.null(names(answer)))
6     names(answer) <- X
7   if (!identical(simplify, FALSE) && length(answer))
8     simplify2array(answer, higher = (simplify == "array"))
9   else answer
10 }
  
```

Console:

```

1 new dataset using a collection of simple and useful functions. Taking the time to
2 do this upfront can save you time and frustration later on in your analysis.
...
=====| 100%
1) would you like to receive credit for completing this course on coursera.org?
2) YES
3) NO
Selection:
  
```

Environment:

Object	Class
ok	Function ()
visualinfo	Function ()

Help Window: apply

Apply a Function Over a Ragged Array

Description
Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Usage
apply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

Arguments
X an atomic object, typically a vector
INDEX list of one or more factors, each of same length as X. The elements are coerced to factors by as.factor

