| Please choose a lesson, or type 0 to return to course menu.

1: Basic Building Blocks 2: Workspace and Files 3: Sequences of Numbers

4: Vectors 5: Missing Values 6: Subsetting Vectors

7: Matrices and Data Frames 8: Logic 9: Functions

10: lapply and sapply 11: vapply and tapply 12: Looking at Data

13: Simulation 14: Dates and Times 15: Base Graphics

Selection: 2

| | 0%

| In this lesson, you'll learn how to examine your local workspace in R and begin to explore the

| relationship between your workspace and the file system of your machine.

...

|== | 2%

| Because different operating systems have different conventions with regards to things like file

| paths, the outputs of these commands may vary across machines.

...

|===== | 5%

| However it's important to note that R provides a common API (a common set of commands) for

| interacting with files, that way your code will work across different kinds of computers.

...

|======= | 7%

| Let's jump right in so you can get a feel for how these special functions work!

...

|========= | 10%

| Determine which directory your R session is using as its current working directory using getwd().

> getwd()

[1] "C:/Users/Ruiz/Documents/Informática/CURSOS/The Data Science Specialization/Swirl"

| That's the answer I was looking for.

|=========== | 12%

| List all the objects in your local workspace using ls().

> ls()

character(0)

| You got it!

|============== | 15%

| Some R commands are the same as their equivalents commands on Linux or on a Mac. Both Linux and Mac

| operating systems are based on an operating system called Unix. It's always a good idea to learn more

| about Unix!

...

|================ | 17%

| Assign 9 to x using x <- 9.

> x <- 9

| That's a job well done!

|================== | 20%

| Now take a look at objects that are in your workspace using ls().

> ls()

[1] "x"

| Excellent job!

|===================== | 22%

| List all the files in your working directory using list.files() or dir().

> dir()

[1] "1. Basic-Building-Blocks.docx" "prueba.RData"

| Excellent job!

|======================= | 24%

| As we go through this lesson, you should be examining the help page for each new function. Check out

| the help page for list.files with the command ?list.files.

> ?list.files

| Keep up the great work!

|========================= | 27%

| One of the most helpful parts of any R help file is the See Also section. Read that section for

| list.files. Some of these functions may be used in later portions of this lesson.

...

|============================ | 29%

| Using the args() function on a function name is also a handy way to see what arguments a function can

| take.

...

|============================== | 32%

| Use the args() function to determine the arguments to list.files().

> args(list.files)

function (path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE,

recursive = FALSE, ignore.case = FALSE, include.dirs = FALSE,

no.. = FALSE)

NULL

| Nice work!

|================================ | 34%

| Assign the value of the current working directory to a variable called "old.dir".

> old.dir <- getwd()

| Nice work!

|================================== | 37%

| We will use old.dir at the end of this lesson to move back to the place that we started. A lot of

| query functions like getwd() have the useful property that they return the answer to the question as

| a result of the function.

...

|===================================== | 39%

| Use dir.create() to create a directory in the current working directory called "testdir".

> dir.create("testdir")

| Excellent job!

|======================================= | 41%

| We will do all our work in this new directory and then delete it after we are done. This is the R

| analog to "Take only pictures, leave only footprints."

...

|========================================= | 44%

| Set your working directory to "testdir" with the setwd() command.

> setwd("testdir")

| You nailed it! Good job!

|============================================ | 46%

| In general, you will want your working directory to be someplace sensible, perhaps created for the

| specific project that you are working on. In fact, organizing your work in R packages using RStudio

| is an excellent option. Check out RStudio at http://www.rstudio.com/

...

|============================================== | 49%

| Create a file in your working directory called "mytest.R" using the file.create() function.

> file.create("mytest.R")

[1] TRUE

| Excellent work!

|================================================ | 51%

| This should be the only file in this newly created directory. Let's check this by listing all the

| files in the current directory.

>

> list.files()

[1] "mytest.R"

| All that hard work is paying off!

|================================================== | 54%

| Check to see if "mytest.R" exists in the working directory using the file.exists() function.

> file.exists("mytest.R")

[1] TRUE

| You got it!

|===================================================== | 56%

| These sorts of functions are excessive for interactive use. But, if you are running a program that

| loops through a series of files and does some processing on each one, you will want to check to see

| that each exists before you try to process it.

...

|======================================================= | 59%

| Access information about the file "mytest.R" by using file.info().

> file.info("mytest.R")

size isdir mode mtime ctime atime exe

mytest.R 0 FALSE 666 2016-01-03 20:55:40 2016-01-03 20:55:40 2016-01-03 20:55:40 no

| Keep up the great work!

|========================================================= | 61%

| You can use the $ operator --- e.g., file.info("mytest.R")$mode --- to grab specific items.

...

|============================================================ | 63%

| Change the name of the file "mytest.R" to "mytest2.R" by using file.rename().

> file.rename("mytest.R", "mytest2.R")

[1] TRUE

| You are really on a roll!

|============================================================== | 66%

| Your operating system will provide simpler tools for these sorts of tasks, but having the ability to

| manipulate files programatically is useful. You might now try to delete mytest.R using

| file.remove('mytest.R'), but that won't work since mytest.R no longer exists. You have already

| renamed it.

...

|================================================================ | 68%

| Make a copy of "mytest2.R" called "mytest3.R" using file.copy().

> file.copy("mytest2.R", "mytest3.R")

[1] TRUE

| That's a job well done!

|================================================================== | 71%

| You now have two files in the current directory. That may not seem very interesting. But what if you

| were working with dozens, or millions, of individual files? In that case, being able to

| programatically act on many files would be absolutely necessary. Don't forget that you can,

| temporarily, leave the lesson by typing play() and then return by typing nxt().

...

|===================================================================== | 73%

| Provide the relative path to the file "mytest3.R" by using file.path().

> file.path("mytest3.R")

[1] "mytest3.R"

| Perseverance, that's the answer.

|======================================================================= | 76%

| You can use file.path to construct file and directory paths that are independent of the operating

| system your R code is running on. Pass 'folder1' and 'folder2' as arguments to file.path to make a

| platform-independent pathname.

> file.path("folder1", "folder2")

[1] "folder1/folder2"

| Keep working like that and you'll get there!

|========================================================================= | 78%

| Take a look at the documentation for dir.create by entering ?dir.create . Notice the 'recursive'

| argument. In order to create nested directories, 'recursive' must be set to TRUE.

> ?dir.create

| You are doing so well!

|============================================================================ | 80%

| Create a directory in the current working directory called "testdir2" and a subdirectory for it

| called "testdir3", all in one command by using dir.create() and file.path().

> dir.create(file.path("testdir2", "testdir3"), recursive = TRUE)

| Nice work!

|============================================================================== | 83%

| To delete a directory you need to use the recursive = TRUE argument with the function unlink(). If

| you don't use recursive = TRUE, R is concerned that you're unaware that you're deleting a directory

| and all of its contents. R reasons that, if you don't specify that recursive equals TRUE, you don't

| know that something is in the directory you're trying to delete. R tries to prevent you from making a

| mistake.

...

|================================================================================ | 85%

| Delete the "testdir2" directory that you created by using unlink().

> unlink("testdir2", recursive = TRUE)

| All that hard work is paying off!

|=================================================================================== | 88%

| Why is this command named "unlink" rather than something more sensible like "dir.delete" or

| "dir.remove"? Mainly, history. unlink is the traditional Unix command for removing directories.

...

|===================================================================================== | 90%

| Go back to your original working directory using setwd(). (Recall that we created the variable

| old.dir with the full path for the orginal working directory at the start of these questions.)

> setwd(old.dir)

| You nailed it! Good job!

|======================================================================================= | 93%

| It is often helpful to save the settings that you had before you began an analysis and then go back

| to them at the end. This trick is often used within functions; you save, say, the par() settings that

| you started with, mess around a bunch, and then set them back to the original values at the end. This

| isn't the same as what we have done here, but it seems similar enough to mention.

...

|========================================================================================= | 95%

| Delete the 'testdir' directory that you just left (and everything in it)

> unlink("testdir", recursive = TRUE)

| You nailed it! Good job!

|============================================================================================ | 98%

| Take nothing but results. Leave nothing but assumptions. That sounds like 'Take nothing but pictures.

| Leave nothing but footprints.' But it makes no sense! Surely our readers can come up with a better

| motto . . .

...

|==============================================================================================| 100%

| 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!